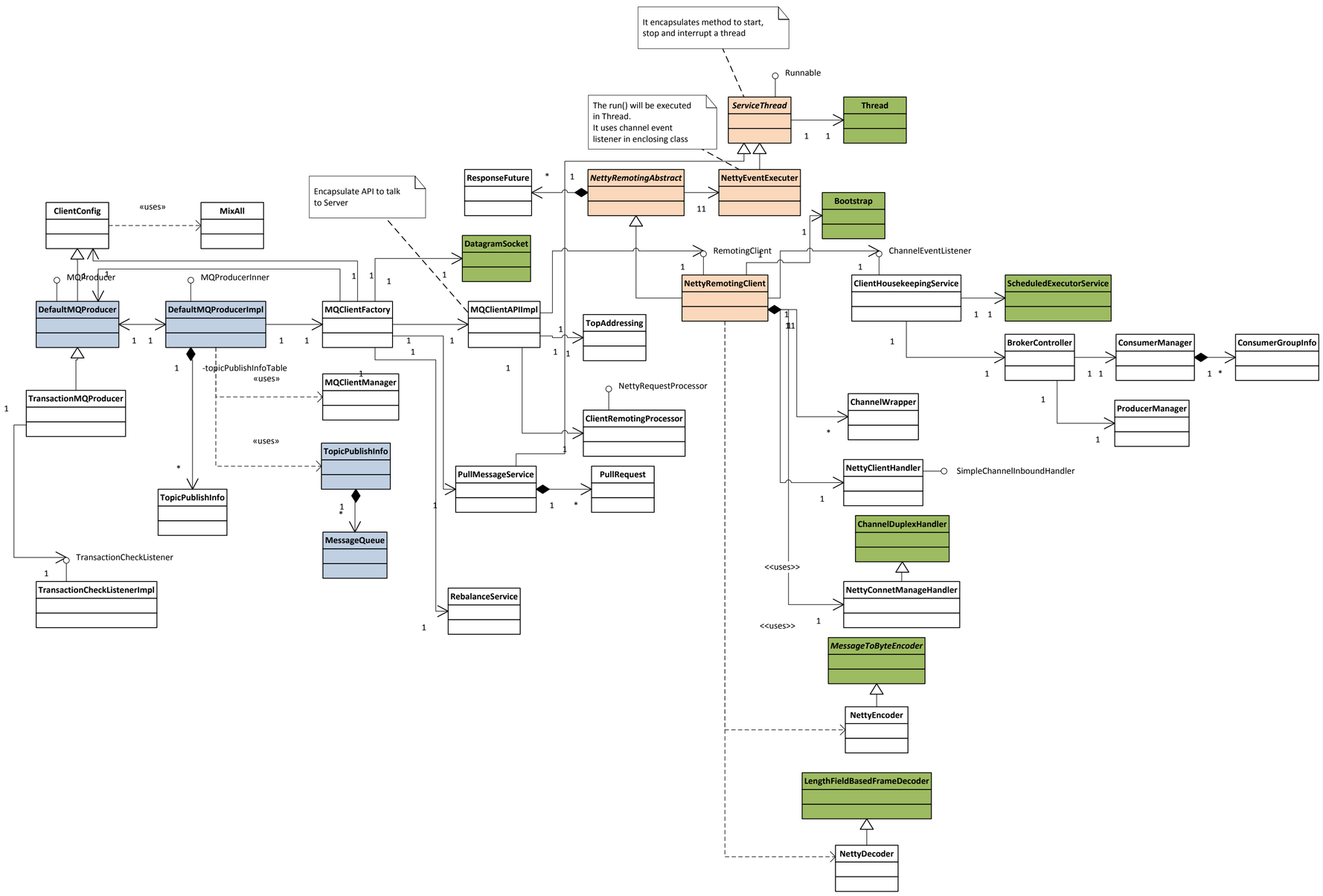
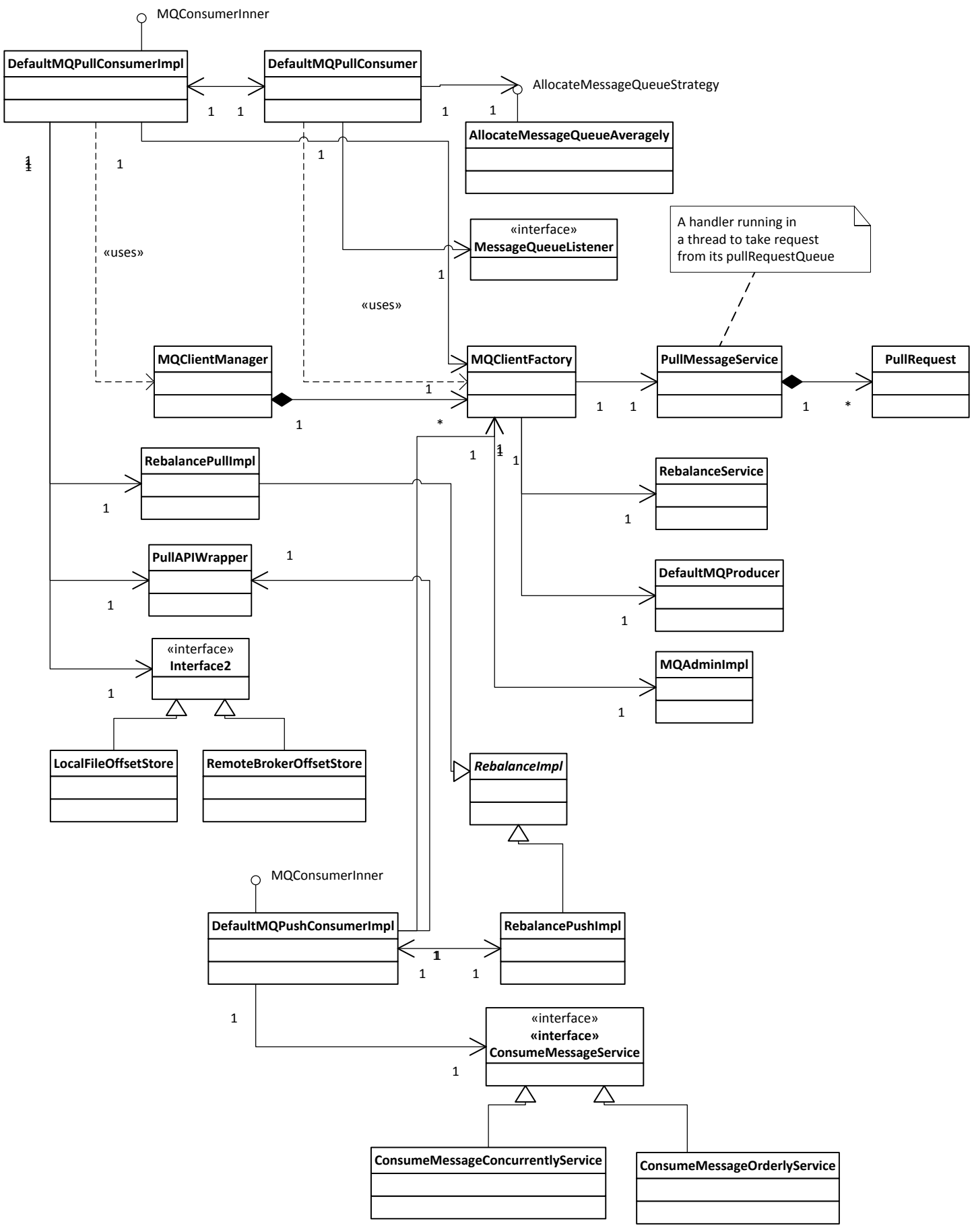
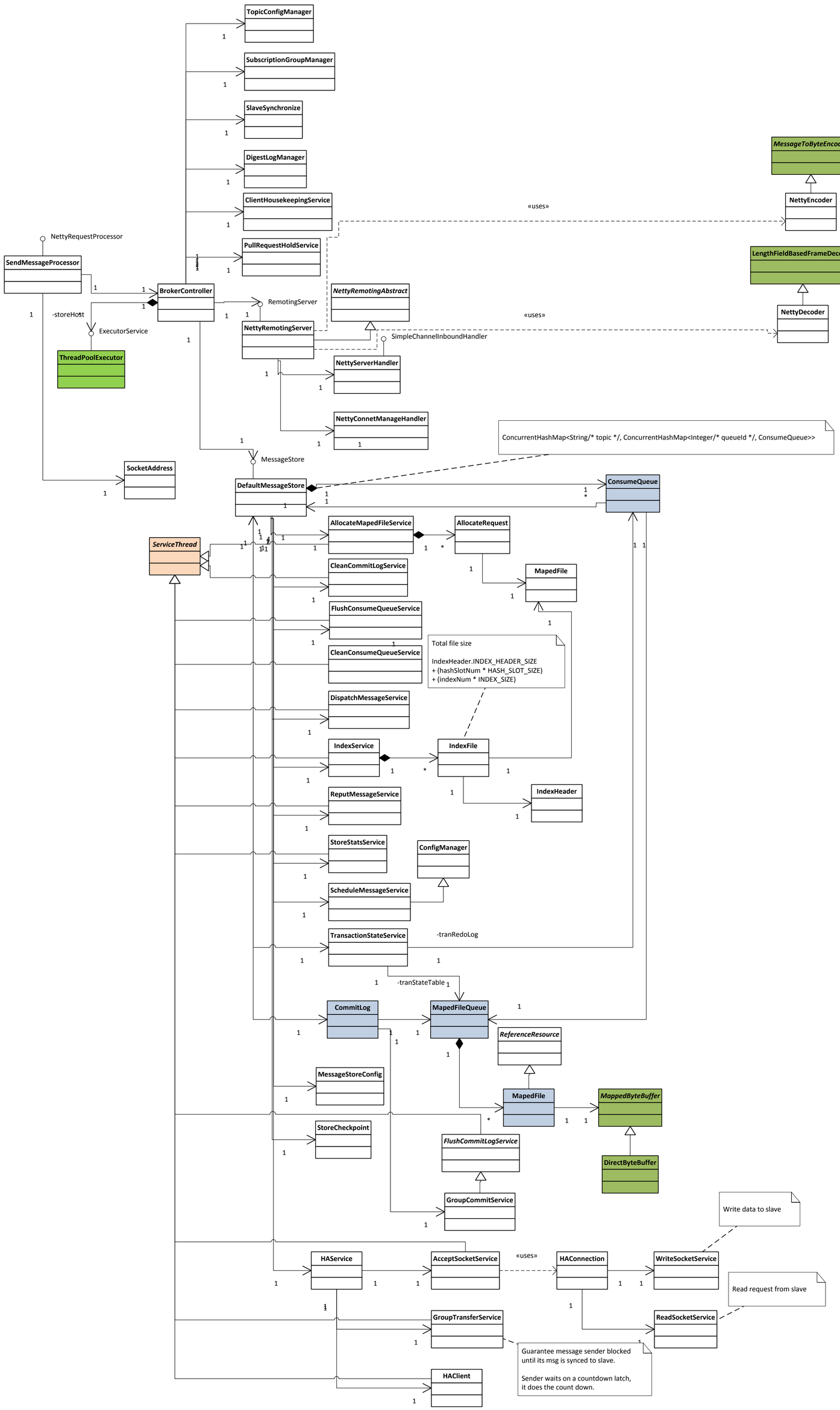
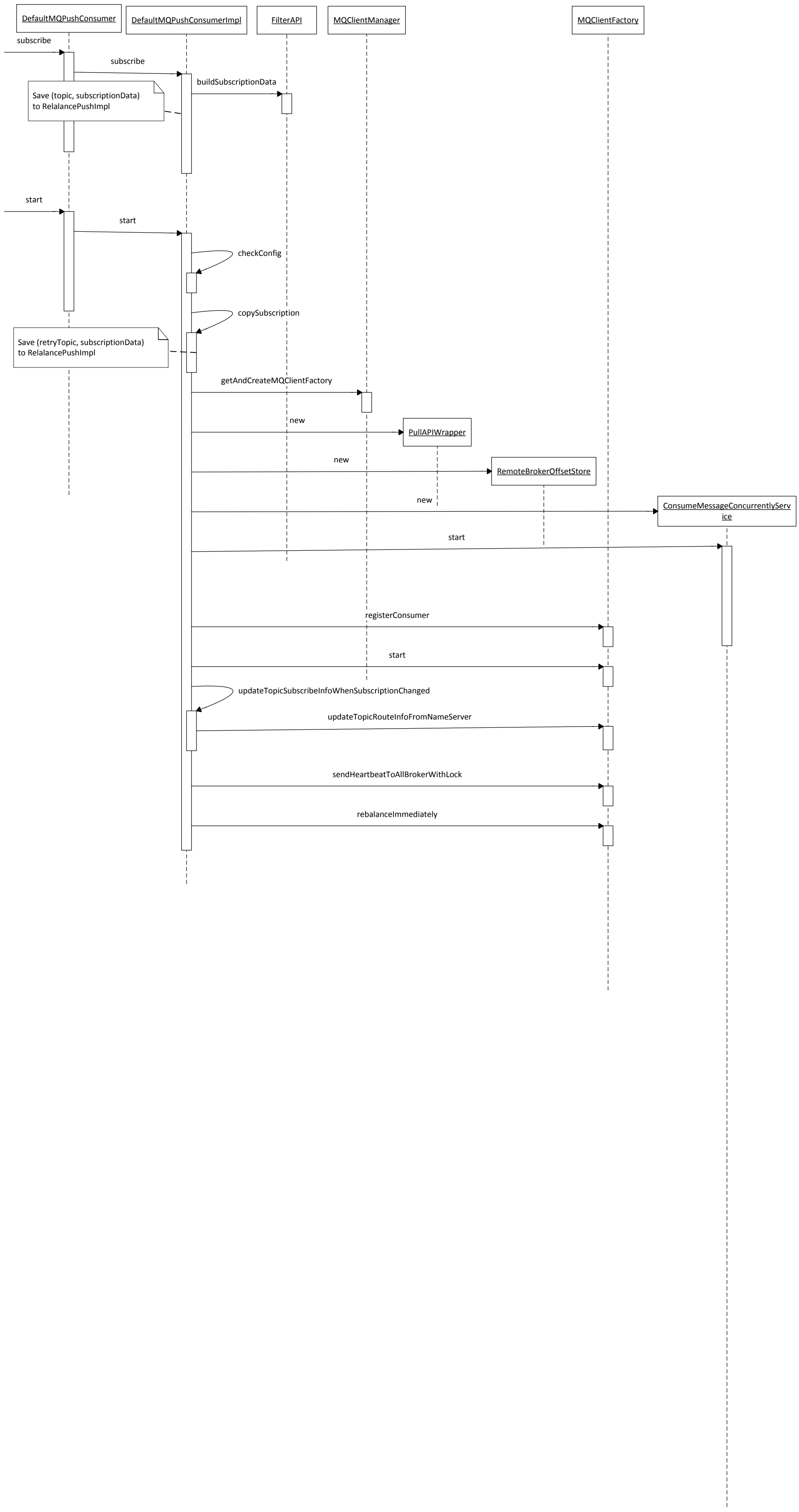
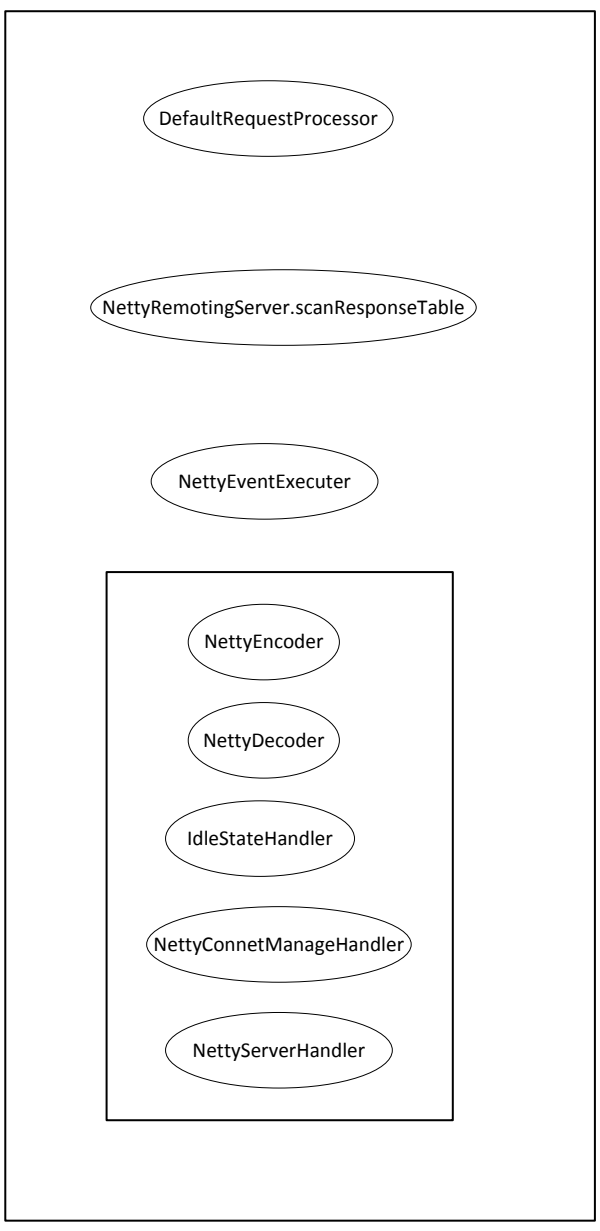
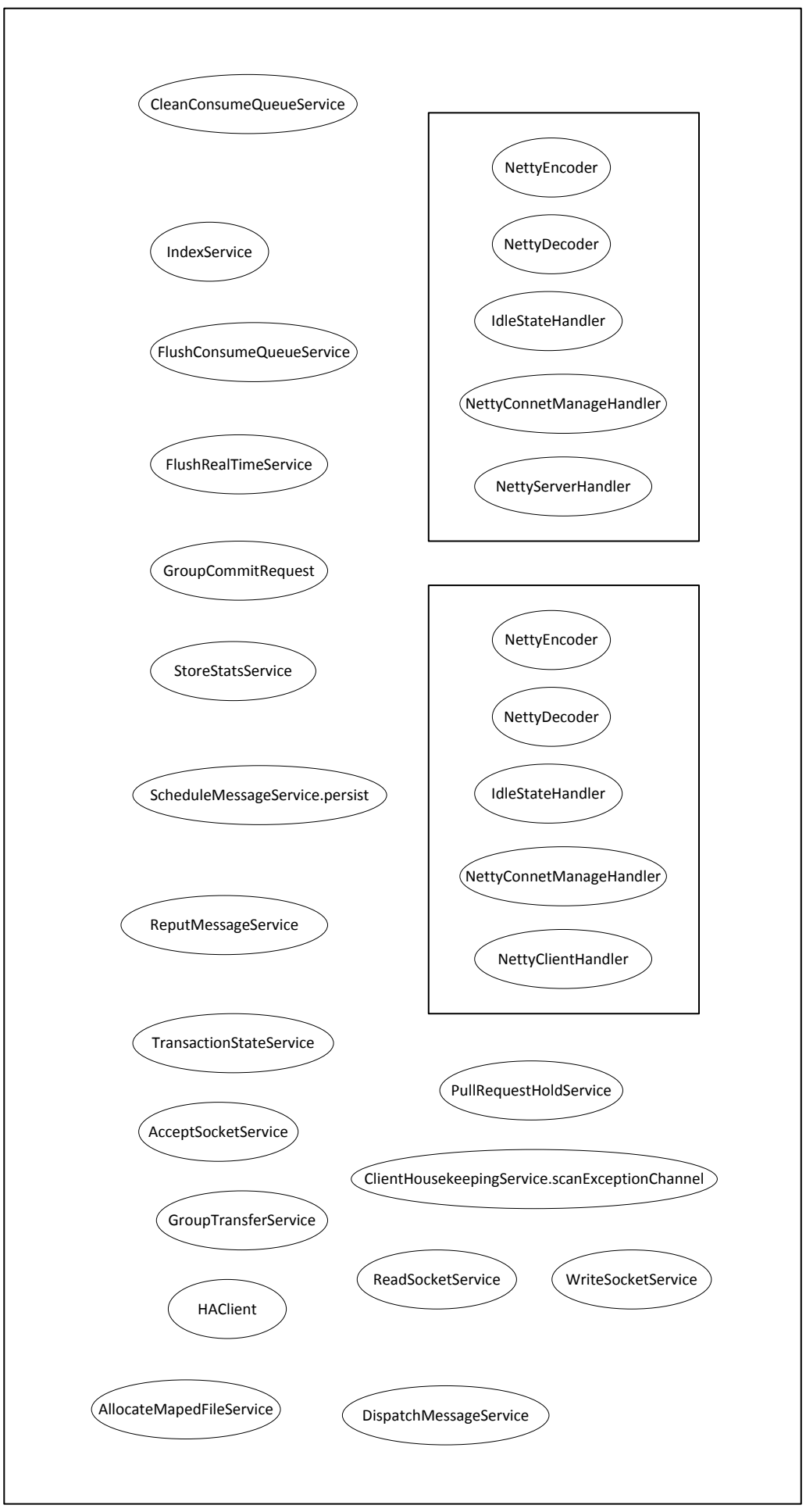

It encapsulates method to start, stop and interrupt a thread
Runnable
The run() will be executed in Thread. It uses channel event listener in enclosing class
ServiceThread
Thread
Encapsulate API to talk to Server
ResponseFuture
NettyRemotingAbstract
NettyEventExecuter
Bootstrap
ClientConfig
«uses»
MixAll
DatagramSocket
RemotingClient
ChannelEventListener
MQProducer
MQProducerInner
NettyRemotingClient
ClientHousekeepingService
ScheduledExecutorService
DefaultMQProducer
DefaultMQProducerImpl
MQClientFactory
MQClientAPIImpl
TopAddressing
BrokerController
ConsumerManager
ConsumerGroupInfo
-topicPublishInfoTable
«uses»
MQClientManager
NettyRequestProcessor
TransactionMQProducer
ClientRemotingProcessor
ChannelWrapper
ProducerManager
«uses»
TopicPublishInfo
PullMessageService
PullRequest
NettyClientHandler
SimpleChannelInboundHandler
TopicPublishInfo
MessageQueue
ChannelDuplexHandler
TransactionCheckListener
<<uses>>
NettyConnetManageHandler
TransactionCheckListenerImpl
RebalanceService
<<uses>>
MessageToByteEncoder
NettyEncoder
LengthFieldBasedFrameDecoder
NettyDecoder

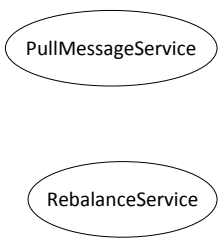

MQConsumerInner
DefaultMQPullConsumerImpl
DefaultMQPullConsumer
AllocateMessageQueueStrategy
AllocateMessageQueueAveragely
«interface»
MessageQueueListener
A handler running in
a thread to take request
from its pullRequestQueue
«uses»
«uses»
MQClientManager
MQClientFactory
PullMessageService
PullRequest
RebalancePullImpl
RebalanceService
PullAPIWrapper
DefaultMQProducer
«interface»
Interface2
MQAdminImpl
LocalFileOffsetStore
RemoteBrokerOffsetStore
RebalanceImpl
MQConsumerInner
DefaultMQPushConsumerImpl
RebalancePushImpl
«interface»
«interface»
ConsumeMessageService
ConsumeMessageConcurrentlyService
ConsumeMessageOrderlyService

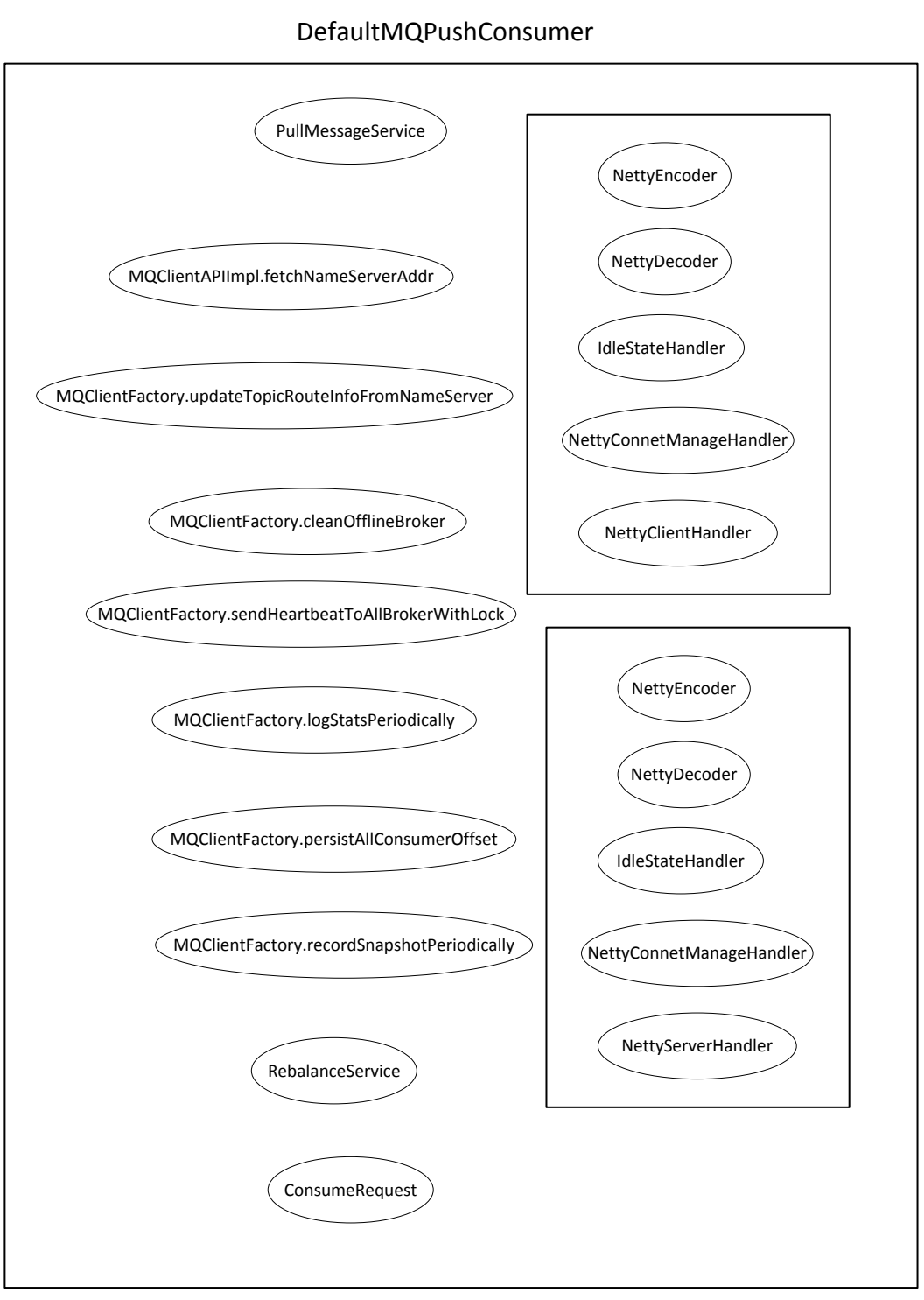

TopicConfigManager
SubscriptionGroupManager
SlaveSynchronize
DigestLogManager
ClientHousekeepingService
PullRequestHoldService
MessageToByteEncoder
NettyEncoder
«uses»
NettyRequestProcessor
SendMessageProcessor
BrokerController
RemotingServer
NettyRemotingAbstract
LengthFieldBasedFrameDecoder
NettyDecoder
«uses»
-storeHost
ExecutorService
ThreadPoolExecutor
NettyRemotingServer
SimpleChannelInboundHandler
NettyServerHandler
NettyConnetManageHandler
ConcurrentHashMap<String/* topic */, ConcurrentHashMap<Integer/* queueId */, ConsumeQueue>>
MessageStore
SocketAddress
DefaultMessageStore
ConsumeQueue
ServiceThread
AllocateMapedFileService
AllocateRequest
CleanCommitLogService
MapedFile
FlushConsumeQueueService
Total file size
IndexHeader.INDEX_HEADER_SIZE
+ (hashSlotNum * HASH_SLOT_SIZE)
+ (indexNum * INDEX_SIZE)
CleanConsumeQueueService
DispatchMessageService
IndexService
IndexFile
IndexHeader
ReputMessageService
StoreStatsService
ConfigManager
ScheduleMessageService
TransactionStateService
-tranRedoLog
-tranStateTable
CommitLog
MapedFileQueue
ReferenceResource
MessageStoreConfig
MapedFile
MappedByteBuffer
StoreCheckpoint
FlushCommitLogService
DirectByteBuffer
GroupCommitService
Write data to slave
HAService
AcceptSocketService
«uses»
HAConnection
WriteSocketService
Read request from slave
GroupTransferService
ReadSocketService
Guarantee message sender blocked
until its msg is synced to slave.
Sender waits on a countdown latch,
it does the count down.
HAClient

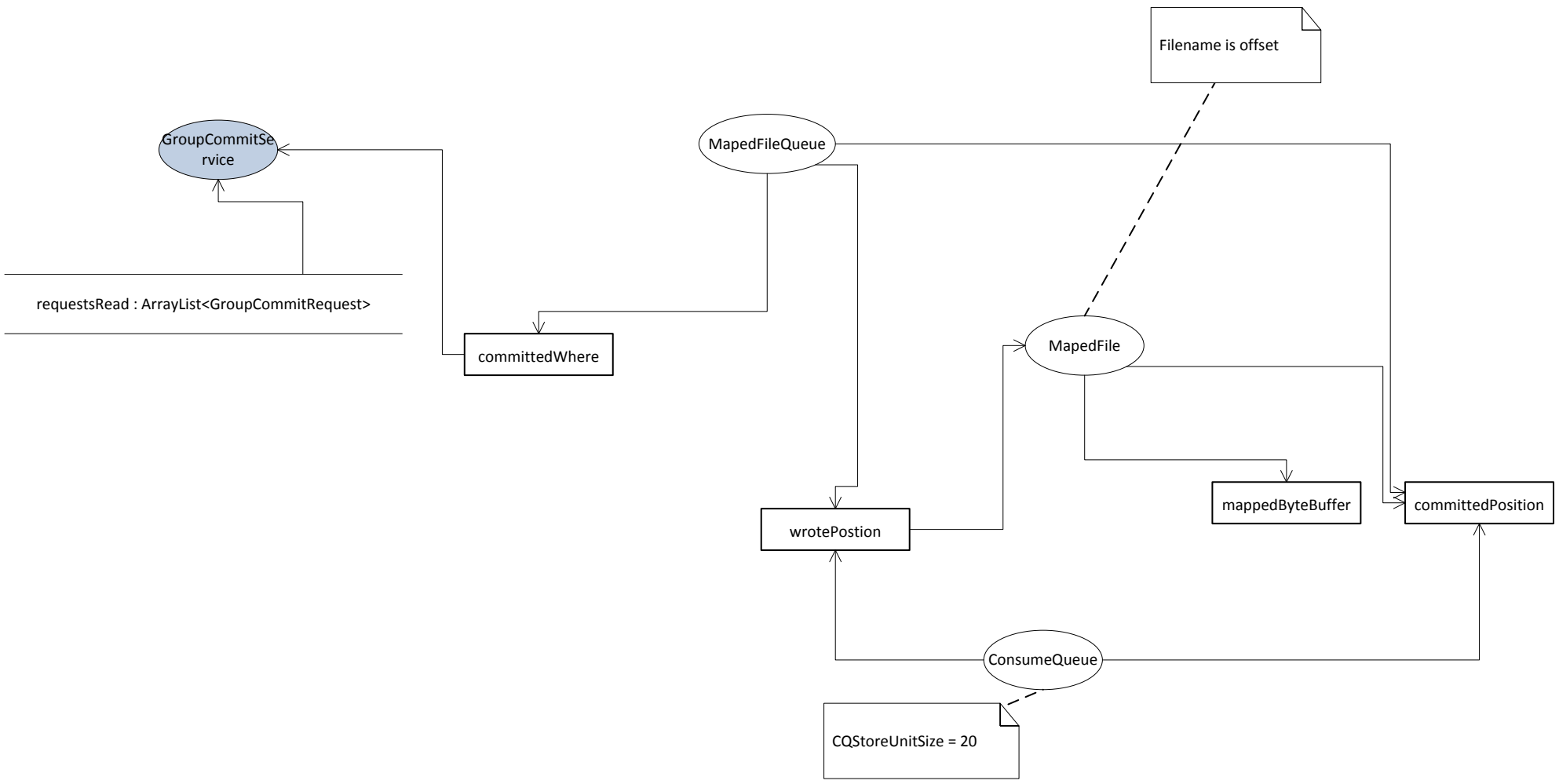

DefaultMQPushConsumer
DefaultMQPushConsumerImpl
FilterAPI
MQClientManager
MQClientFactory
subscribe
subscribe
buildSubscriptionData
Save (topic, subscriptionData)
to RelalancePushImpl
start
start
checkConfig
copySubscription
Save (retryTopic, subscriptionData)
to RelalancePushImpl
getAndCreateMQClientFactory
getAndCreateMQClientFactory
new
PullAPIWrapper
new
RemoteBrokerOffsetStore
new
ConsumeMessageConcurrentlyService
start
registerConsumer
start
updateTopicSubscribeInfoWhenSubscriptionChanged
updateTopicRouteInfoFromNameServer
sendHeartbeatToAllBrokerWithLock
rebalanceImmediately

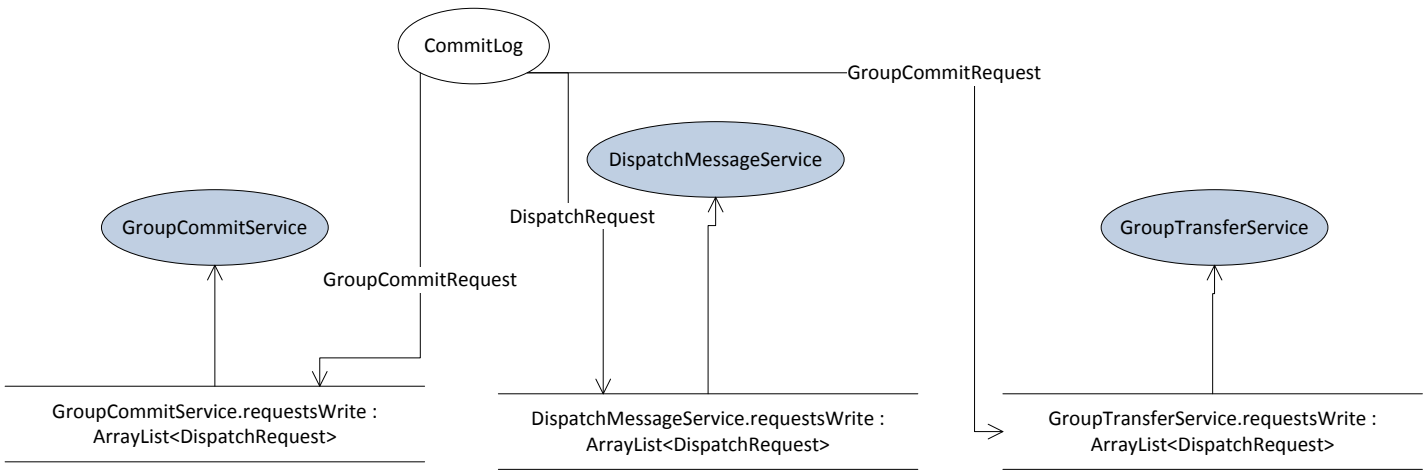

NameServer
DefaultRequestProcessor
NettyRemotingServer.scanResponseTable
NettyEventExecuter
NettyEncoder
NettyDecoder
IdleStateHandler
NettyConnetManageHandler
NettyServerHandler

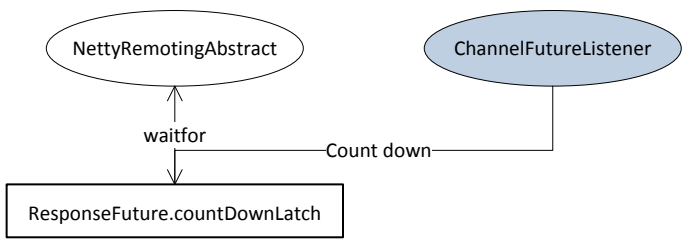

Broker
CleanConsumeQueueService
IndexService
FlushConsumeQueueService
FlushRealTimeService
GroupCommitRequest
StoreStatsService
ScheduleMessageService.persist
ReputMessageService
TransactionStateService
AcceptSocketService
GroupTransferService
HAClient
AllocateMapedFileService
NettyEncoder
NettyDecoder
IdleStateHandler
NettyConnetManageHandler
NettyServerHandler
NettyEncoder
NettyDecoder
IdleStateHandler
NettyConnetManageHandler
NettyClientHandler
PullRequestHoldService
ClientHousekeepingService.scanExceptionChannel
ReadSocketService
WriteSocketService
DispatchMessageService

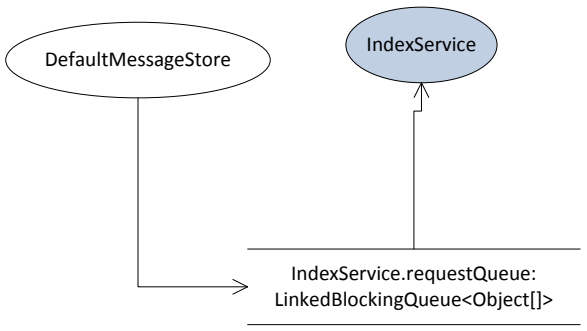

PullMessageService
RebalanceService

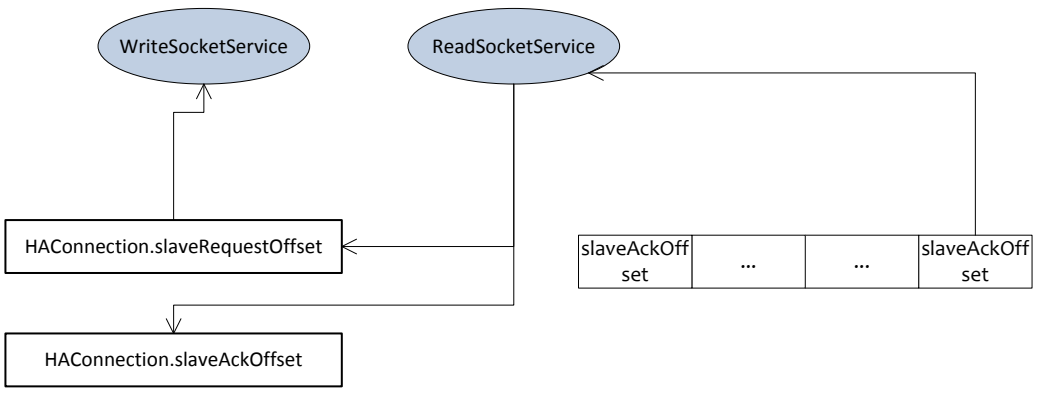

DefaultMQPushConsumer
PullMessageService
MQClientAPIImpl.fetchNameServerAddr
MQClientFactory.updateTopicRouteInfoFromNameServer
MQClientFactory.cleanOfflineBroker
MQClientFactory.sendHeartbeatToAllBrokerWithLock
MQClientFactory.logStatsPeriodically
MQClientFactory.persistAllConsumerOffset
MQClientFactory.recordSnapshotPeriodically
RebalanceService
ConsumeRequest
NettyEncoder
NettyDecoder
IdleStateHandler
NettyConnetManageHandler
NettyClientHandler
NettyEncoder
NettyDecoder
IdleStateHandler
NettyConnetManageHandler
NettyServerHandler

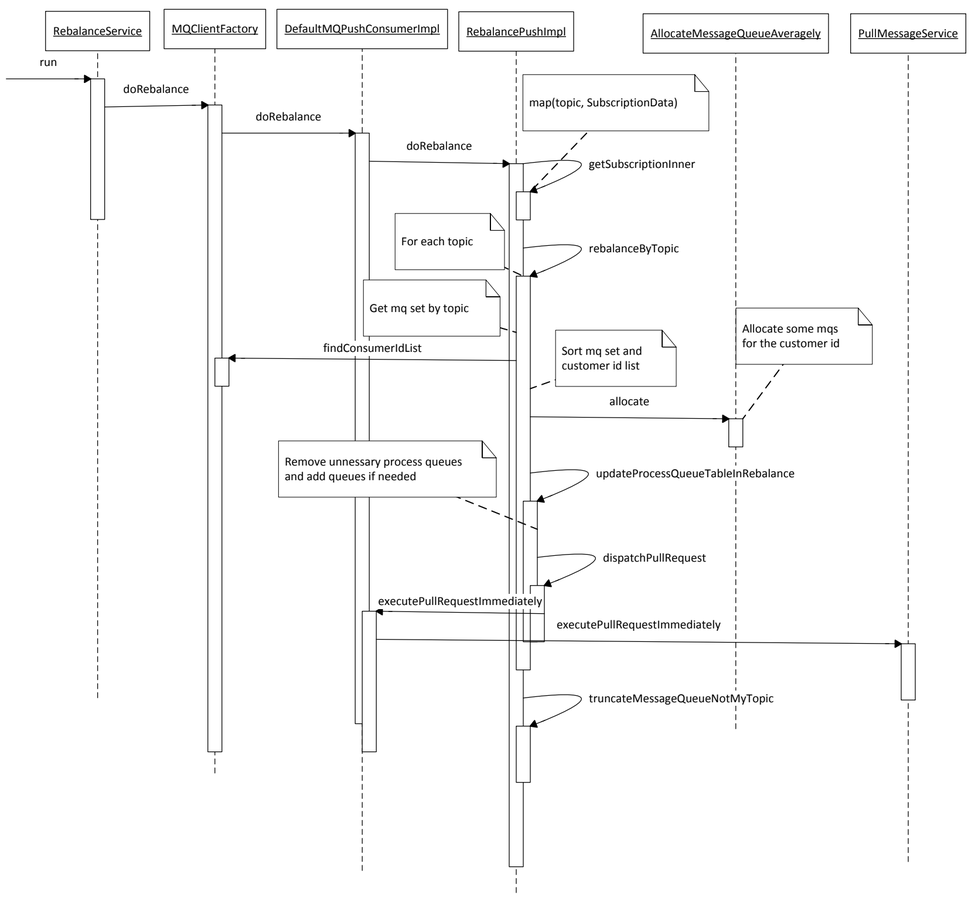

Filename is offset
GroupCommitSe
rvice
MapedFileQueue
requestsRead : ArrayList<GroupCommitRequest>
committedWhere
MapedFile
mappedByteBuffer
committedPosition
wrotePostion
ConsumeQueue
CQStoreUnitSize = 20

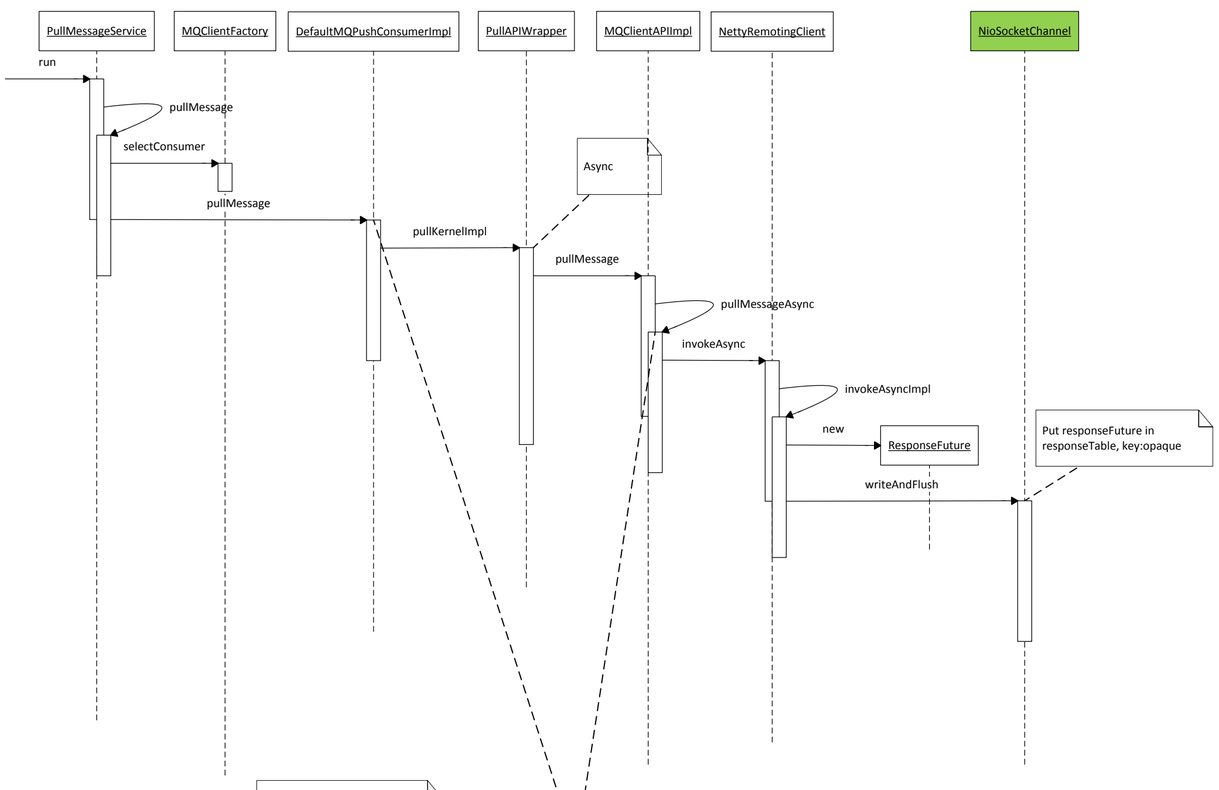

CommitLog
GroupCommitRequest
DispatchMessageService
DispatchRequest
GroupCommitService
GroupTransferService
GroupCommitRequest
GroupCommitService.requestsWrite :
ArrayList<DispatchRequest>
DispatchMessageService.requestsWrite :
ArrayList<DispatchRequest>
GroupTransferService.requestsWrite :
ArrayList<DispatchRequest>

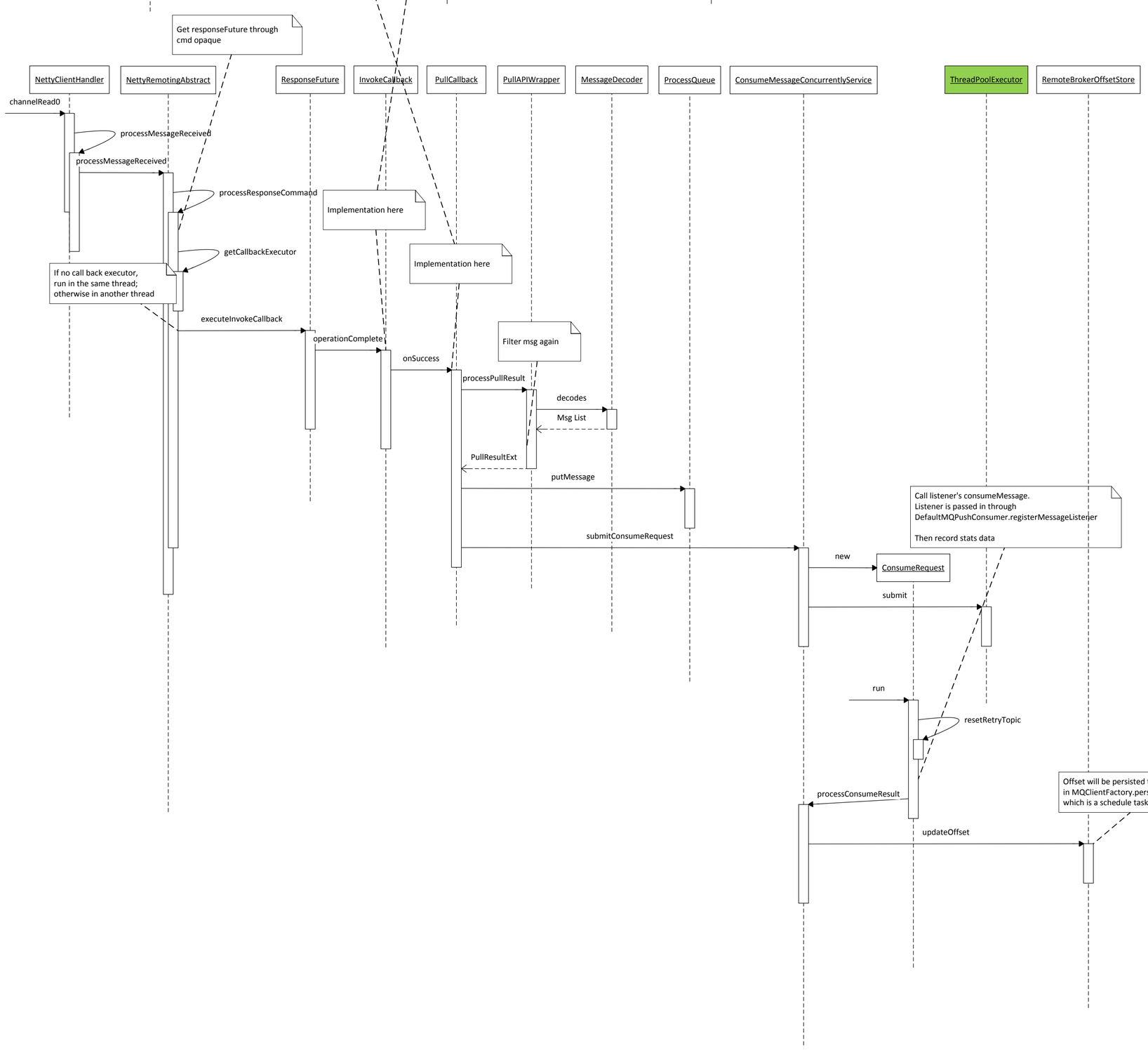

NettyRemotingAbstract
ChannelFutureListener
waitfor
Count down
ResponseFuture.countDownLatch

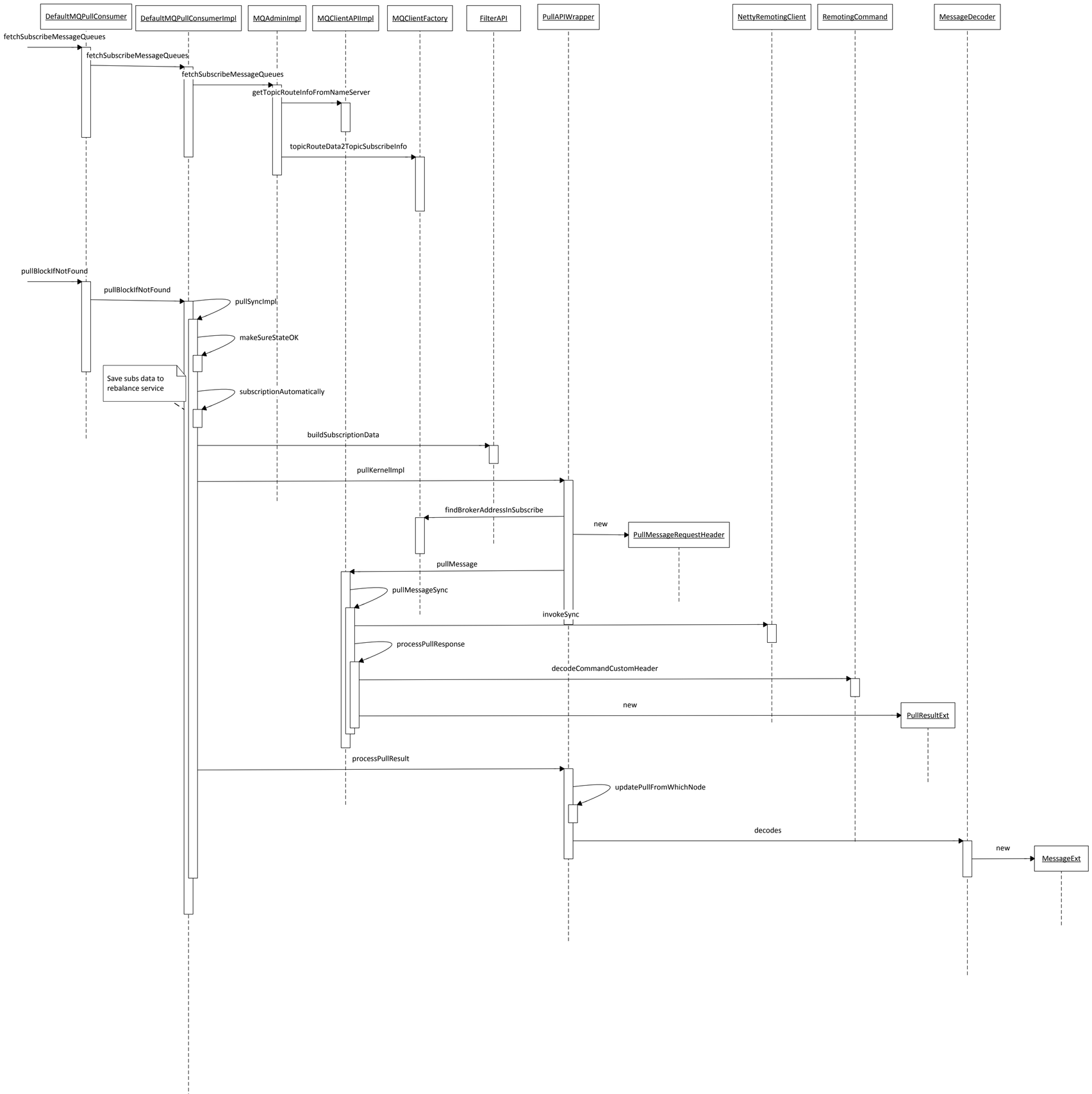

DefaultMessageStore
IndexService
IndexService.requestQueue:
LinkedBlockingQueue<Object[]>

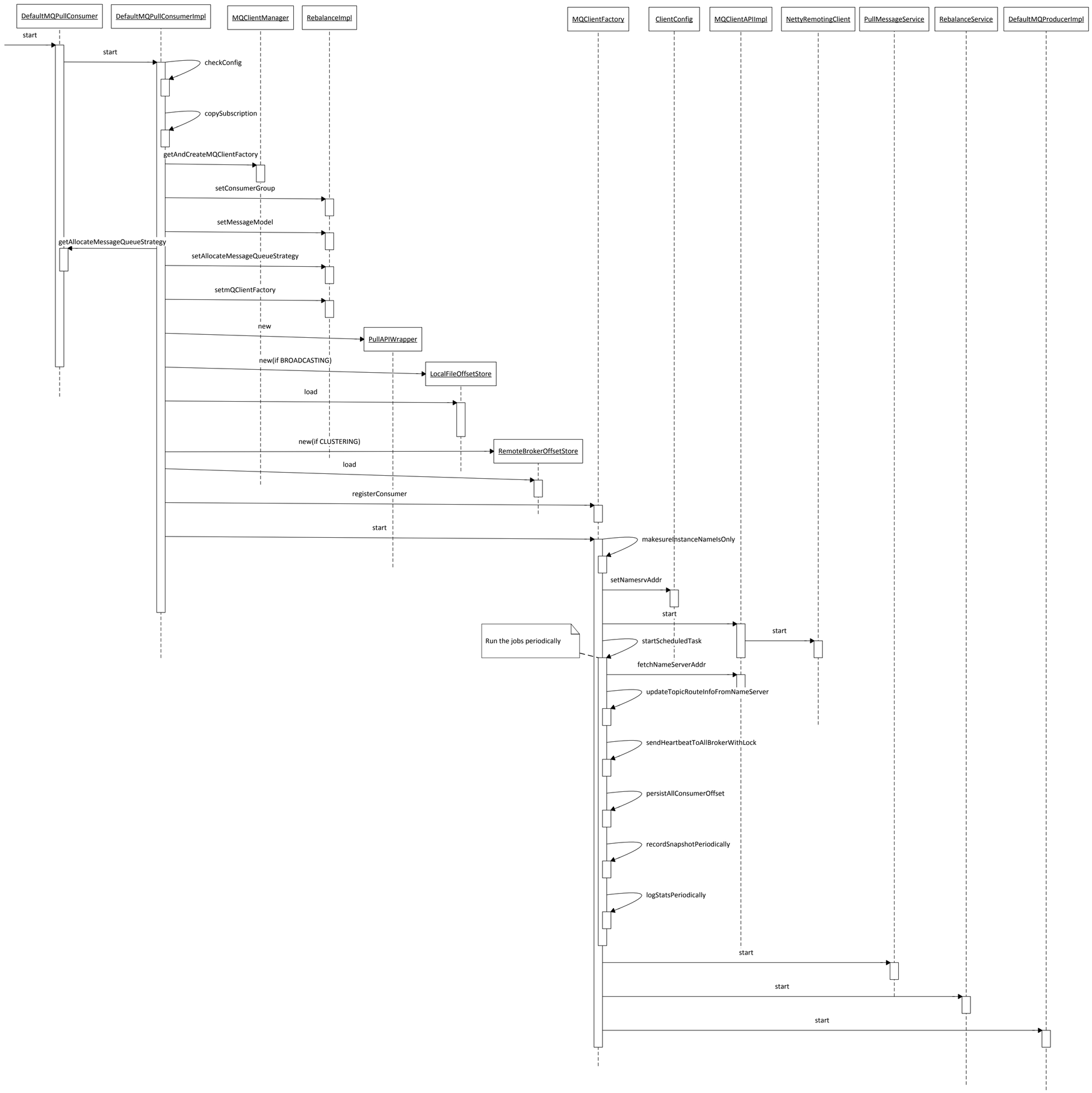

WriteSocketService
ReadSocketService
HAConnection.slaveRequestOffset
slaveAckOff
set
...
...
slaveAckOff
set
HAConnection.slaveAckOffset

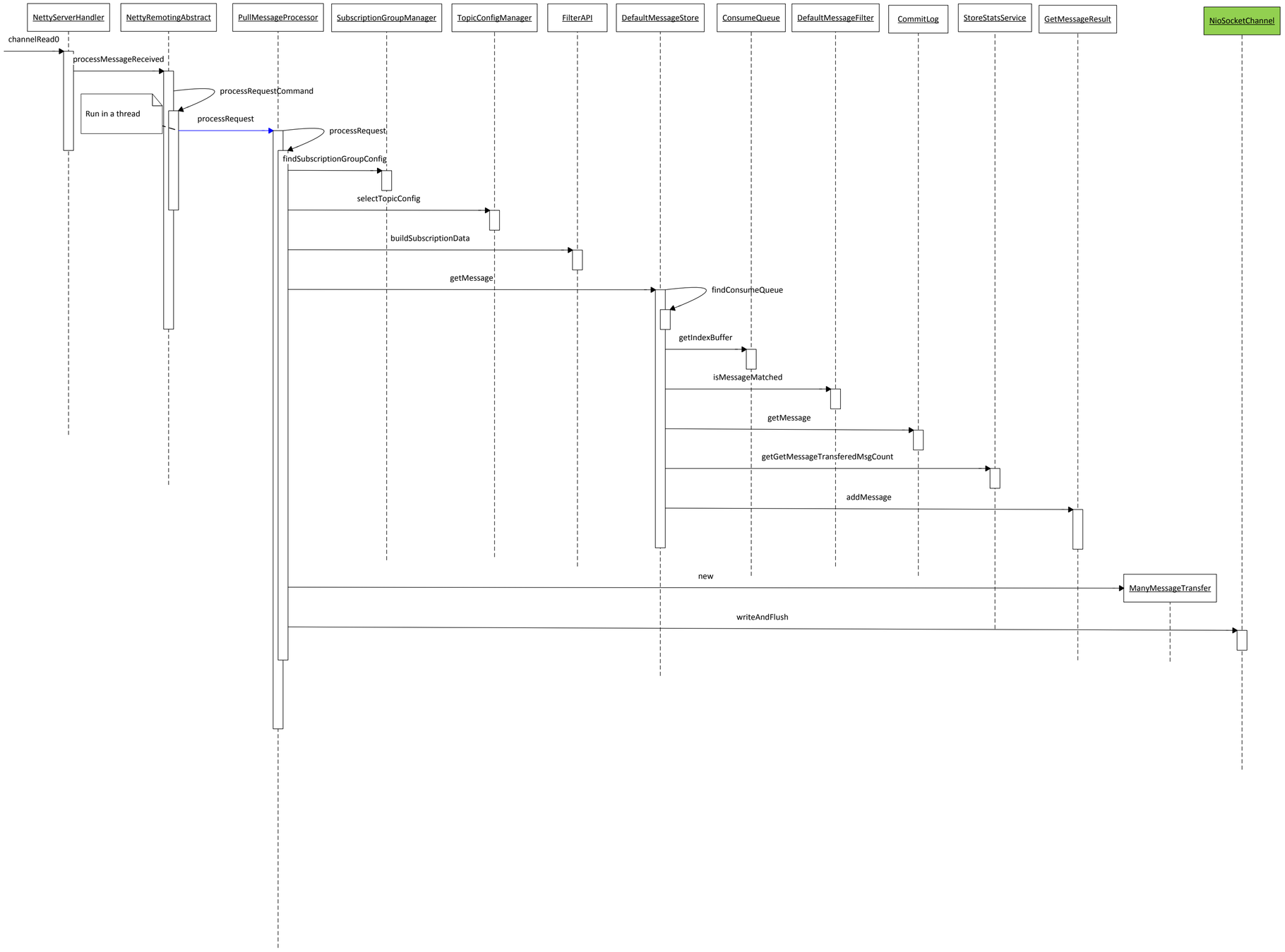

RebalanceService
MQClientFactory
DefaultMQPushConsumerImpl
RebalancePushImpl
AllocateMessageQueueAveragely
PullMessageService
run
doRebalance
doRebalance
doRebalance
map(topic, SubscriptionData)
getSubscriptionInner
For each topic
rebalanceByTopic
Get mq set by topic
Allocate some mqs
for the customer id
findConsumerIdList
Sort mq set and
customer id list
allocate
Remove unnessary process queues
and add queues if needed
updateProcessQueueTableInRebalance
dispatchPullRequest
executePullRequestImmediately
executePullRequestImmediately
truncateMessageQueueNotMyTopic

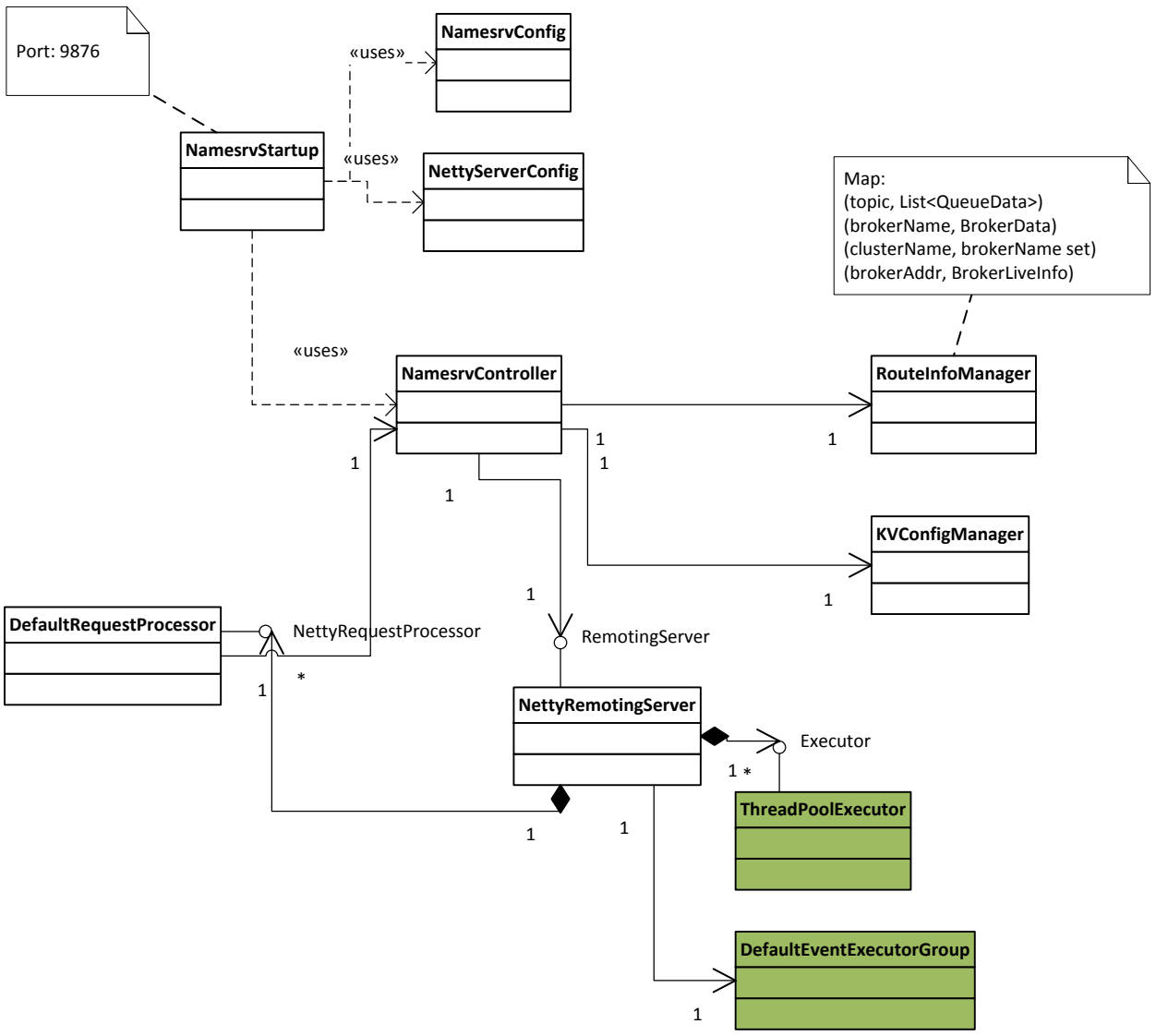

PullMessageService
MQClientFactory
DefaultMQPushConsumerImpl
PullAPIWrapper
MQClientAPIImpl
NettyRemotingClient
NioSocketChannel
run
pullMessage
selectConsumer
Async
pullMessage
pullKernelImpl
pullMessage
pullMessageAsync
invokeAsync
invokeAsyncImpl
new
ResponseFuture
Put responseFuture in
responseTable, key:opaque
writeAndFlush

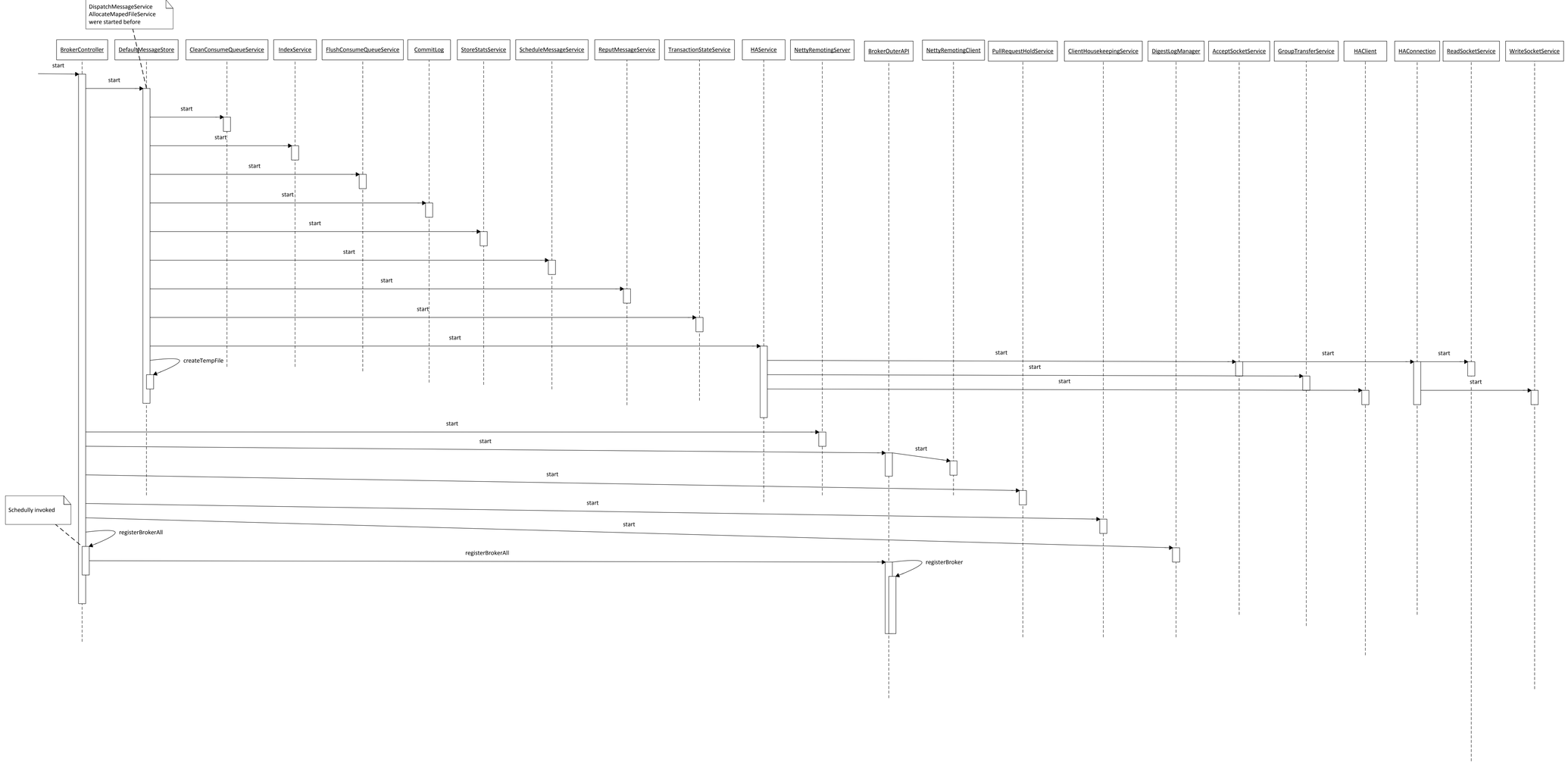

Get responseFuture through
cmd opaque
NettyClientHandler
NettyRemotingAbstract
ResponseFuture
InvokeCallback
PullCallback
PullAPIWrapper
MessageDecoder
ProcessQueue
ConsumeMessageConcurrentlyService
ThreadPoolExecutor
RemoteBrokerOffsetStore
channelRead0
processMessageReceived
processMessageReceived
processResponseCommand
Implementation here
getCallbackExecutor
Implementation here
If no call back executor,
run in the same thread;
otherwise in another thread
executeInvokeCallback
operationComplete
Filter msg again
onSuccess
processPullResult
decodes
Msg List
PullResultExt
putMessage
Call listener's consumeMessage.
Listener is passed in through
DefaultMQPushConsumer.registerMessageListener
Then record stats data
submitConsumeRequest
new
ConsumeRequest
submit
run
resetRetryTopic
Offset will be persisted to broker
in MQClientFactory.persistAllConsumerOffset
which is a schedule task
processConsumeResult
updateOffset

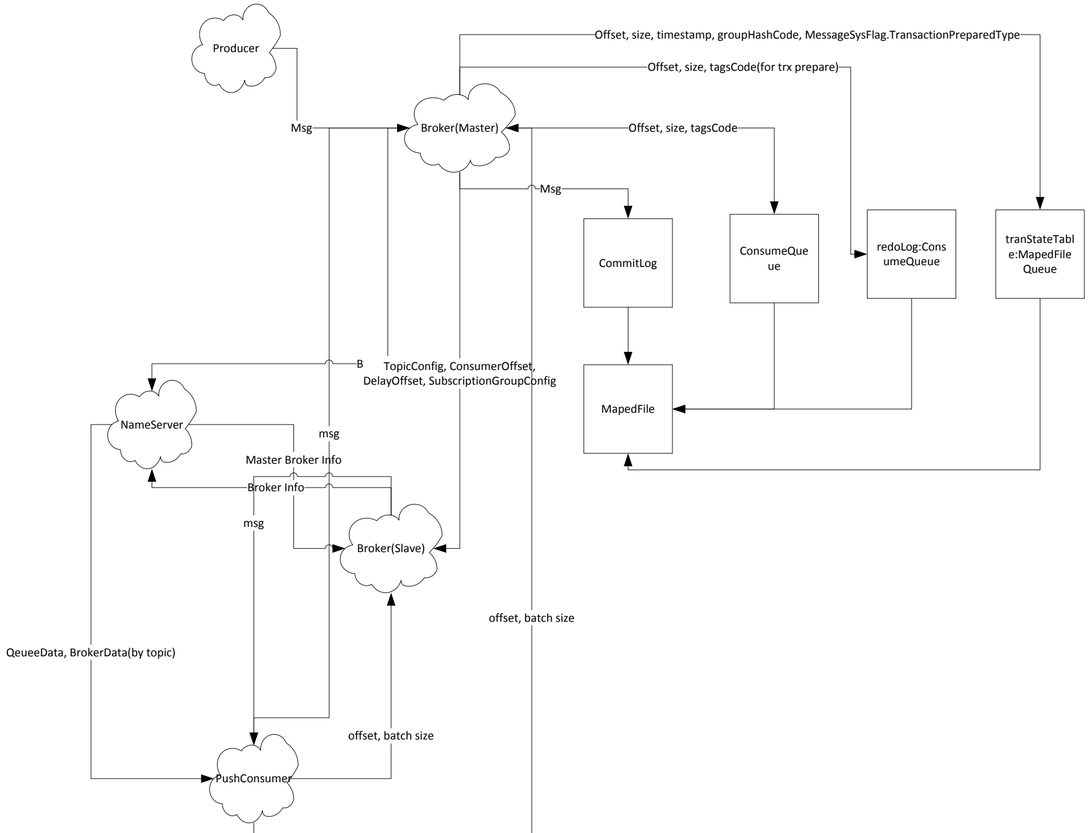

DefaultMQPullConsumer
DefaultMQPullConsumerImpl
MQAdminImpl
MQClientAPIImpl
MQClientFactory
FilterAPI
PullAPIWrapper
NettyRemotingClient
RemotingCommand
MessageDecoder
fetchSubscribeMessageQueues
fetchSubscribeMessageQueues
fetchSubscribeMessageQueues
getTopicRouteInfoFromNameServer
topicRouteData2TopicSubscribeInfo
pullBlockIfNotFound
pullBlockIfNotFound
pullSyncImpl
makeSureStateOK
Save subs data to rebalance service
subscriptionAutomatically
buildSubscriptionData
pullKernelImpl
findBrokerAddressInSubscribe
new
PullMessageRequestHeader
pullMessage
pullMessageSync
invokeSync
processPullResponse
decodeCommandCustomHeader
new
PullResultExt
processPullResult
updatePullFromWhichNode
decodes
new
MessageExt

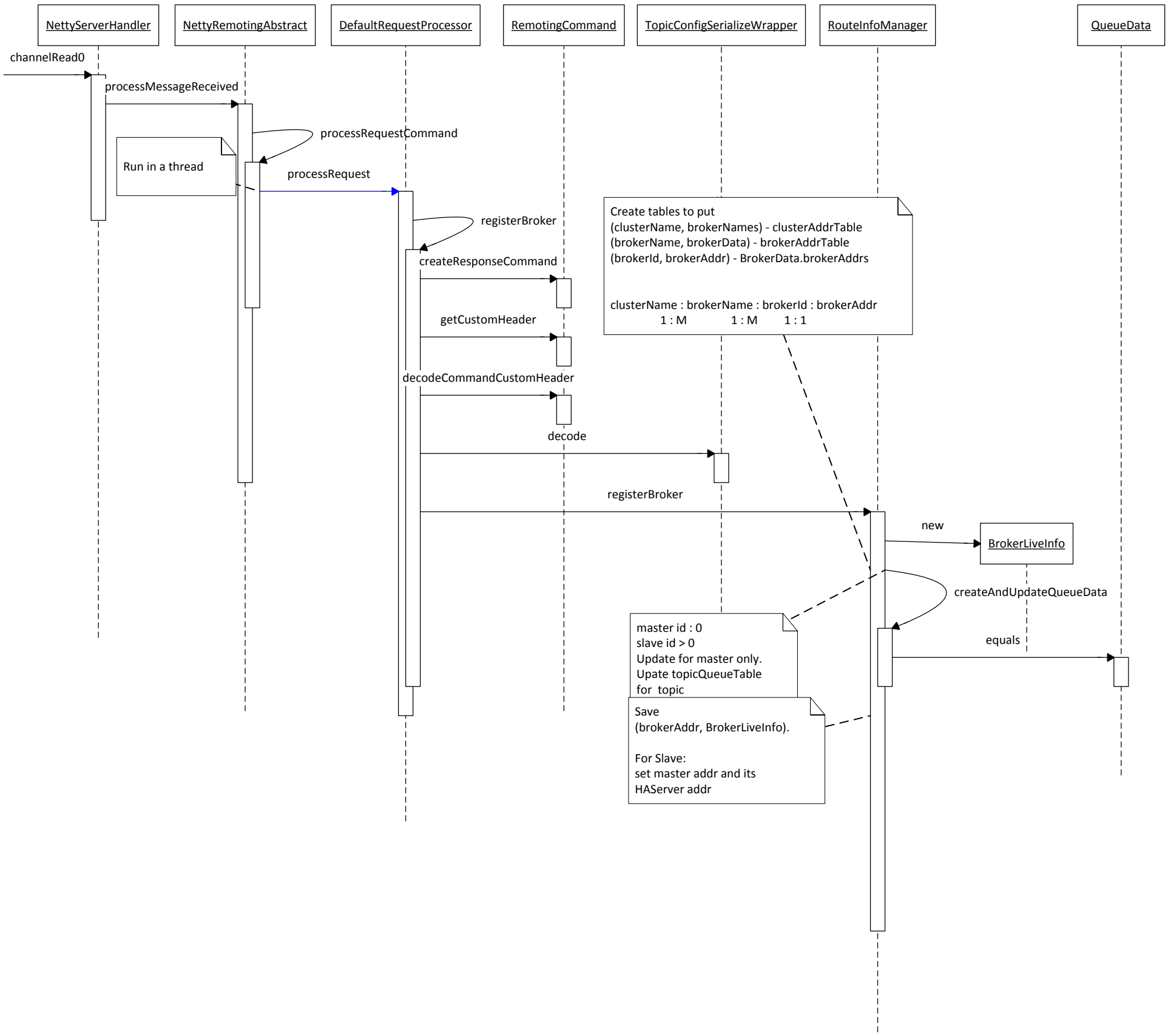

DefaultMQPullConsumer
DefaultMQPullConsumerImpl
MQClientManager
RebalanceImpl
MQClientFactory
ClientConfig
MQClientAPIImpl
NettyRemotingClient
PullMessageService
RebalanceService
DefaultMQProducerImpl
start
start
checkConfig
copySubscription
getAndCreateMQClientFactory
setConsumerGroup
setMessageModel
getAllocateMessageQueueStrategy
setAllocateMessageQueueStrategy
setmQClientFactory
new
PullAPIWrapper
new(if BROADCASTING)
LocalFileOffsetStore
load
new(if CLUSTERING)
RemoteBrokerOffsetStore
load
registerConsumer
start
makesureInstanceNameIsOnly
setNamesrvAddr
start
start
Run the jobs periodically
startScheduledTask
fetchNameServerAddr
updateTopicRouteInfoFromNameServer
sendHeartbeatToAllBrokerWithLock
persistAllConsumerOffset
recordSnapshotPeriodically
logStatsPeriodically
start
start
start

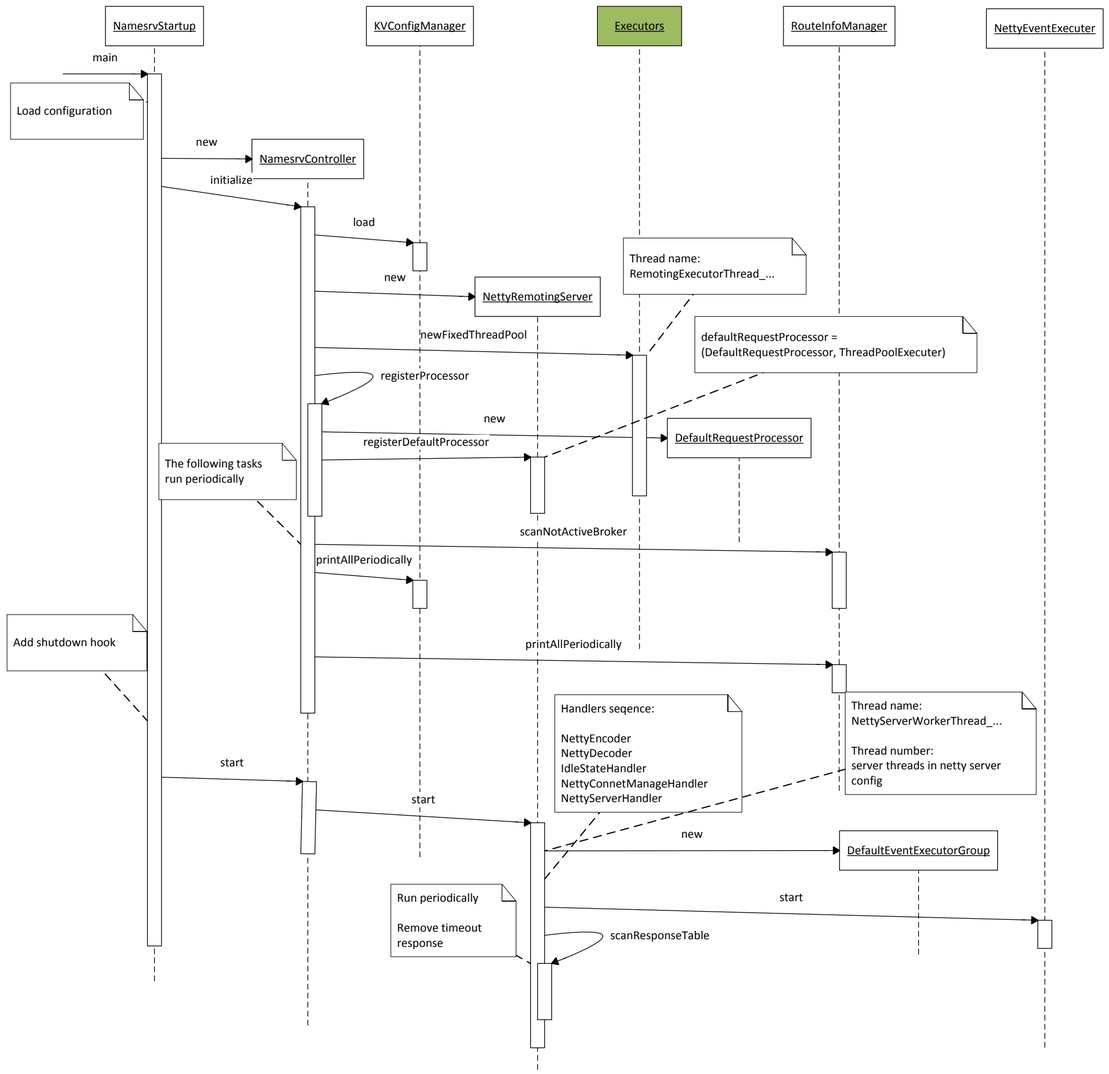

NettyServerHandler
NettyRemotingAbstract
PullMessageProcessor
SubscriptionGroupManager
TopicConfigManager
FilterAPI
DefaultMessageStore
ConsumeQueue
DefaultMessageFilter
CommitLog
StoreStatsService
GetMessageResult
NioSocketChannel
channelRead0
processMessageReceived
processRequestCommand
Run in a thread
processRequest
processRequest
findSubscriptionGroupConfig
selectTopicConfig
buildSubscriptionData
getMessage
findConsumeQueue
getIndexBuffer
isMessageMatched
getMessage
getGetMessageTransferedMsgCount
addMessage
new
ManyMessageTransfer
writeAndFlush

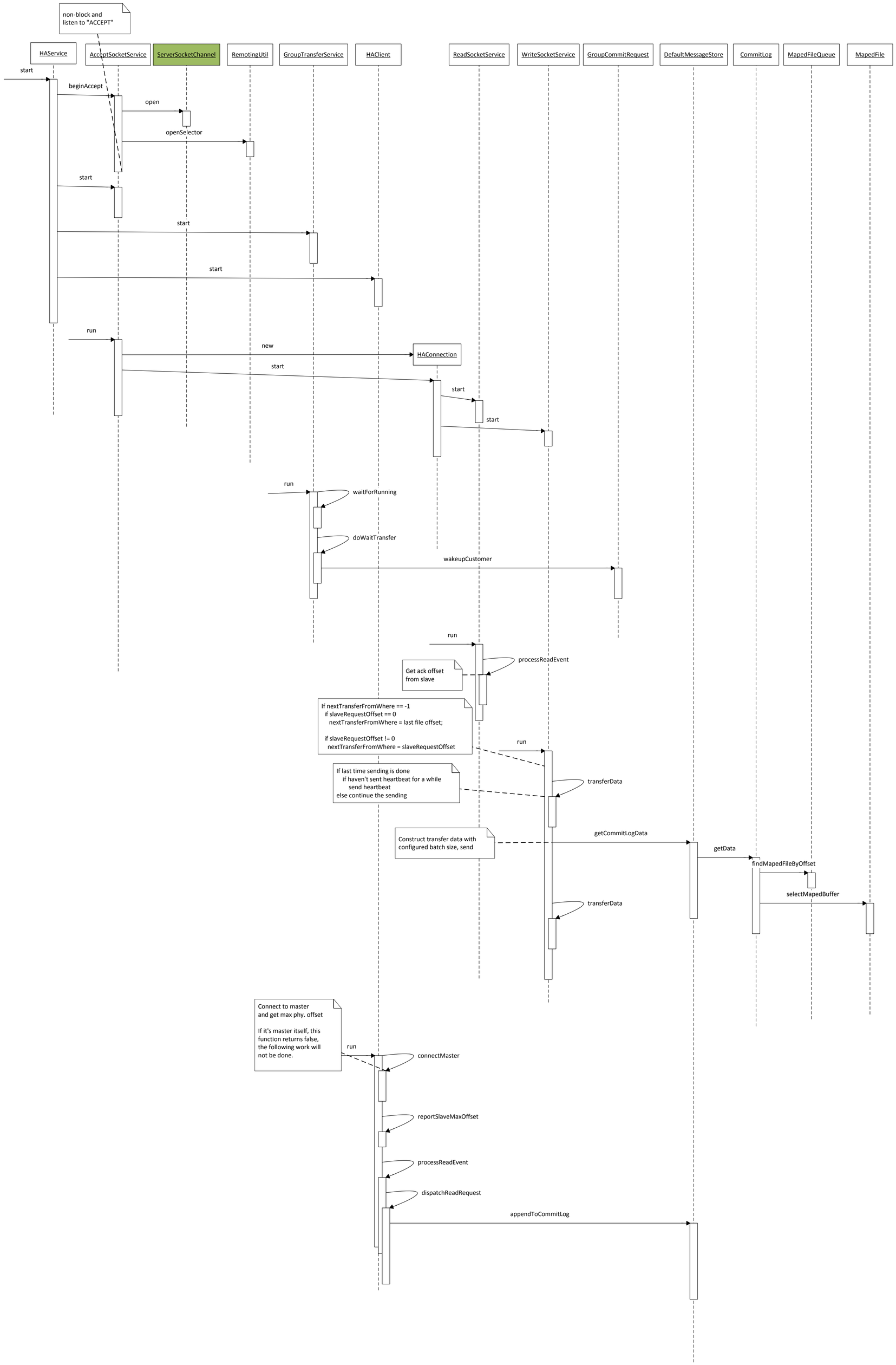

Port: 9876
NamesrvConfig
«uses»
NamesrvStartup
«uses»
NettyServerConfig
Map:
(topic, List<QueueData>)
(brokerName, BrokerData)
(clusterName, brokerName set)
(brokerAddr, BrokerLiveInfo)
«uses»
NamesrvController
RouteInfoManager
1
1
1
1
1
1
KVConfigManager
1
1
DefaultRequestProcessor
NettyRequestProcessor
RemotingServer
1
*
NettyRemotingServer
Executor
1 *
ThreadPoolExecutor
1
1
DefaultEventExecutorGroup
1

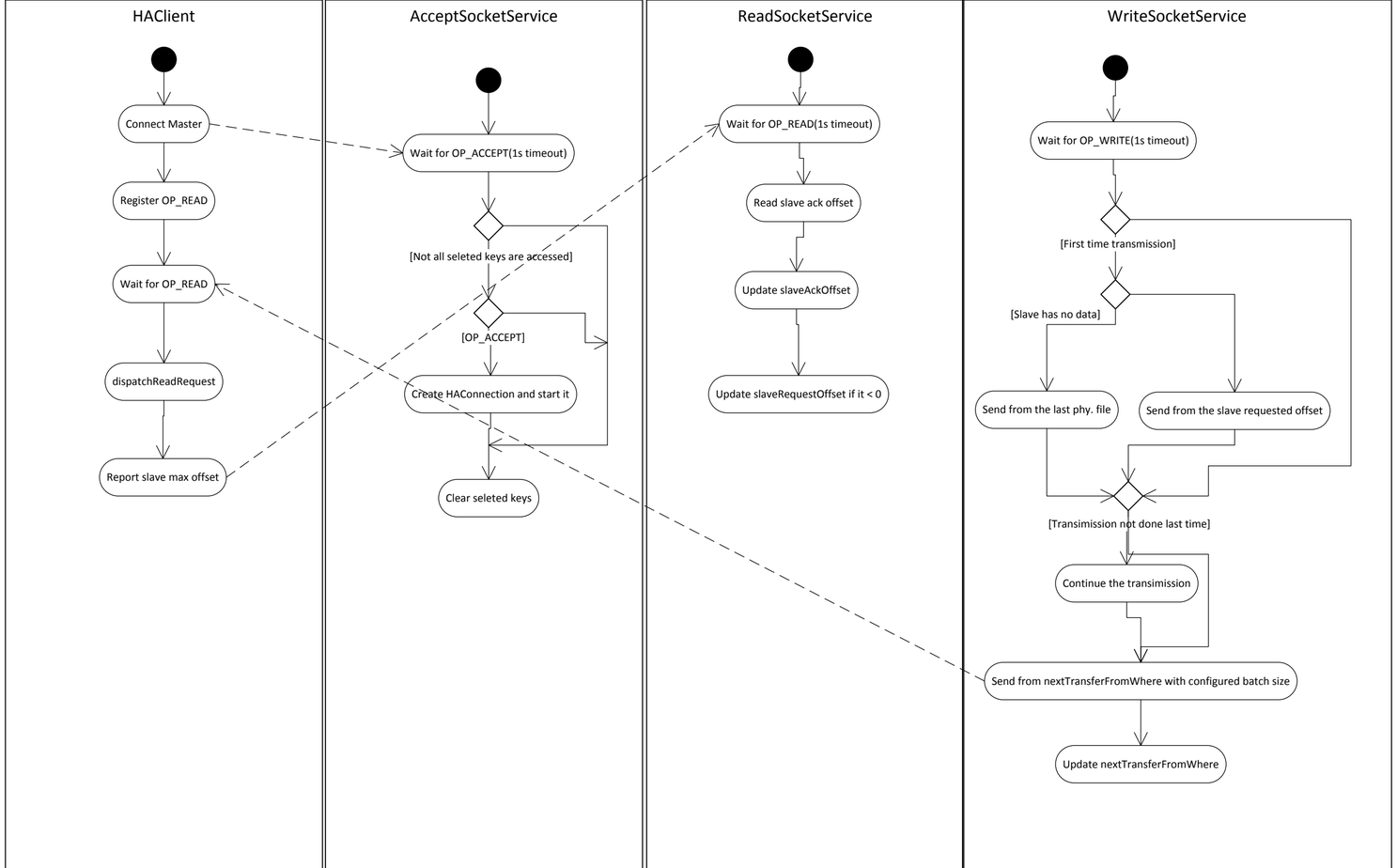

DispatchMessageService
AllocateMapedFileService
were started before
BrokerController
DefaultMessageStore
CleanConsumeQueueService
IndexService
FlushConsumeQueueService
CommitLog
StoreStatsService
ScheduleMessageService
ReputMessageService
TransactionStateService
HAService
NettyRemotingServer
BrokerOuterAPI
NettyRemotingClient
PullRequestHoldService
ClientHousekeepingService
DigestLogManager
AcceptSocketService
GroupTransferService
HAClient
HAConnection
ReadSocketService
WriteSocketService
start
createTempFile
Schedully invoked
registerBrokerAll
registerBroker

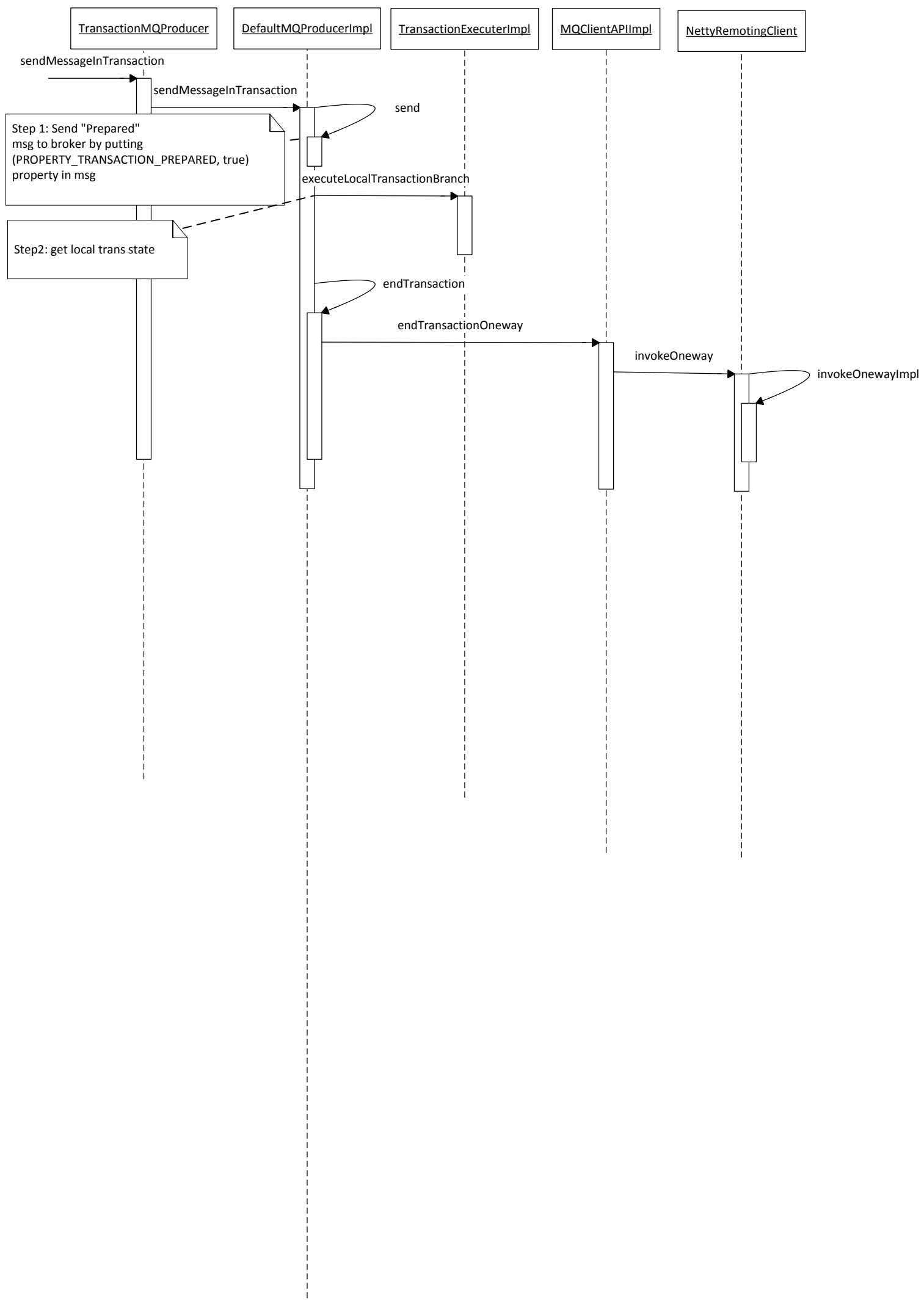

Producer
Offset, size, timestamp, groupHashCode, MessageSysFlag.TransactionPreparedType
Offset, size, tagsCode(for trx prepare)
Msg
Broker(Master)
Offset, size, tagsCode
Msg
CommitLog
ConsumeQue
ue
redoLog:Cons
umeQueue
tranStateTabl
e:MapedFile
Queue
B
TopicConfig, ConsumerOffset,
DelayOffset, SubscriptionGroupConfig
MapedFile
NameServer
msg
Master Broker Info
Broker Info
msg
Broker(Slave)
offset, batch size
QeueeData, BrokerData(by topic)
offset, batch size
PushConsumer

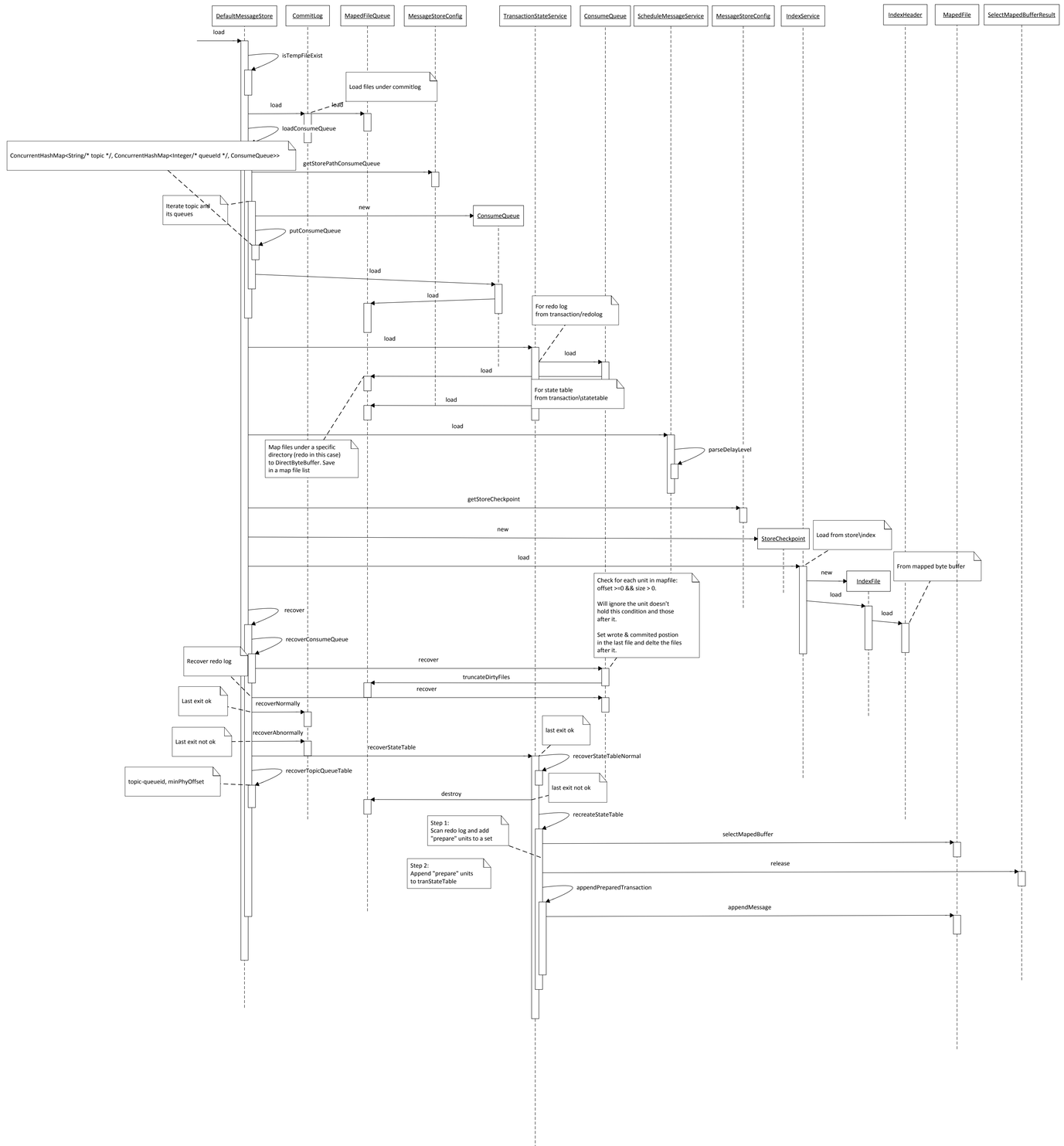

NettyServerHandler
NettyRemotingAbstract
DefaultRequestProcessor
RemotingCommand
TopicConfigSerializeWrapper
RouteInfoManager
QueueData
channelRead0
processMessageReceived
processRequestCommand
Run in a thread
processRequest
registerBroker
Create tables to put
(clusterName, brokerNames) - clusterAddrTable
(brokerName, brokerData) - brokerAddrTable
(brokerId, brokerAddr) - BrokerData.brokerAddrs
clusterName : brokerName : brokerId : brokerAddr
1 : M 1 : M 1 : 1
createResponseCommand
getCustomHeader
decodeCommandCustomHeader
decode
registerBroker
new
BrokerLiveInfo
createAndUpdateQueueData
equals
master id : 0
slave id > 0
Update for master only.
Upate topicQueueTable
for topic
Save
(brokerAddr, BrokerLiveInfo).
For Slave:
set master addr and its
HAServer addr

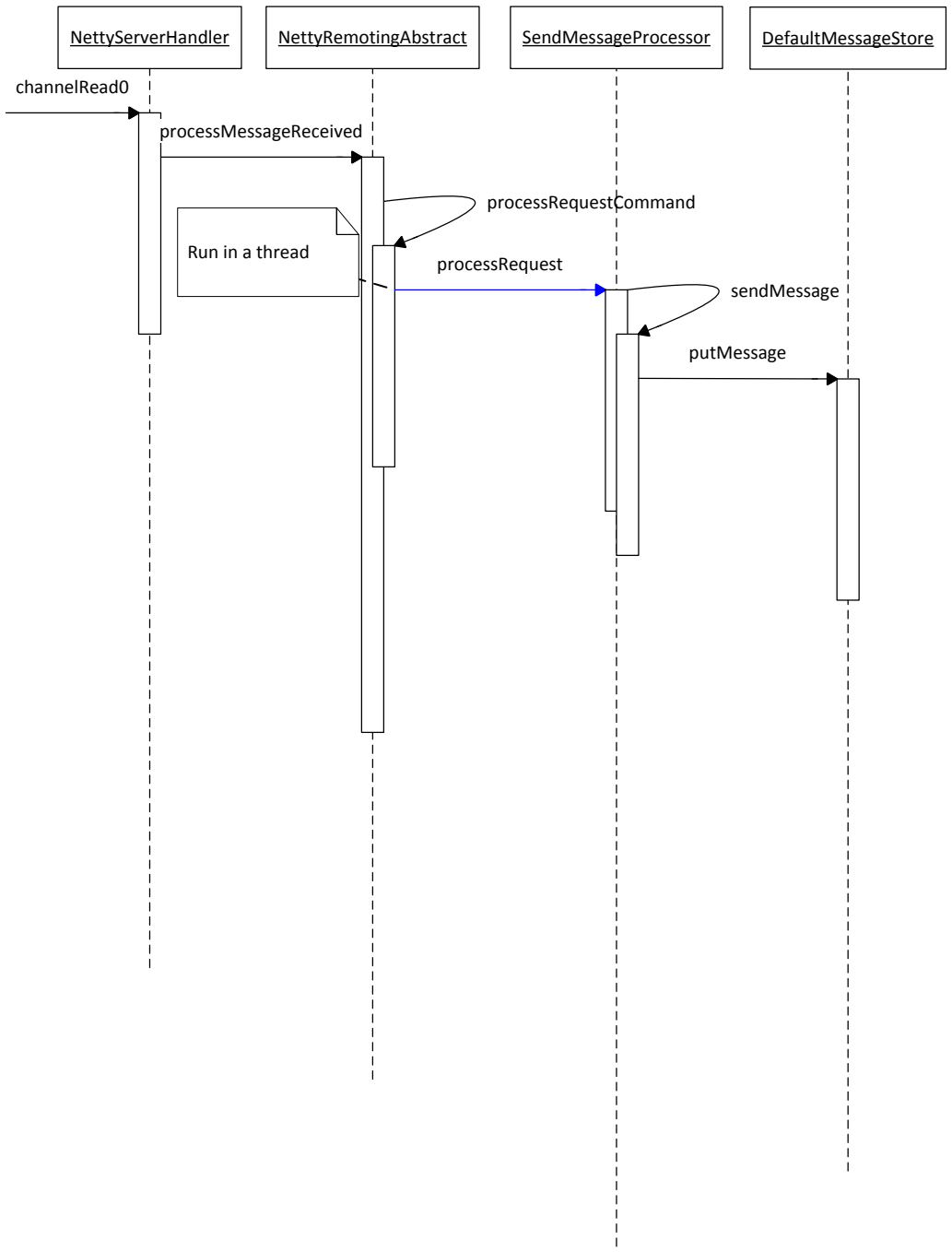

NamesrvStartup
KVConfigManager
Executors
RouteInfoManager
NettyEventExecuter
main
Load configuration
new
NamesrvController
initialize
load
new
NettyRemotingServer
Thread name:
RemotingExecutorThread_...
newFixedThreadPool
defaultRequestProcessor =
(DefaultRequestProcessor, ThreadPoolExecuter)
registerProcessor
new
registerDefaultProcessor
DefaultRequestProcessor
The following tasks
run periodically
scanNotActiveBroker
printAllPeriodically
Add shutdown hook
printAllPeriodically
Handlers seqence:
NettyEncoder
NettyDecoder
IdleStateHandler
NettyConnetManageHandler
NettyServerHandler
Thread name:
NettyServerWorkerThread_...
Thread number:
server threads in netty server
config
start
start
new
DefaultEventExecutorGroup
Run periodically
Remove timeout
response
start
scanResponseTable

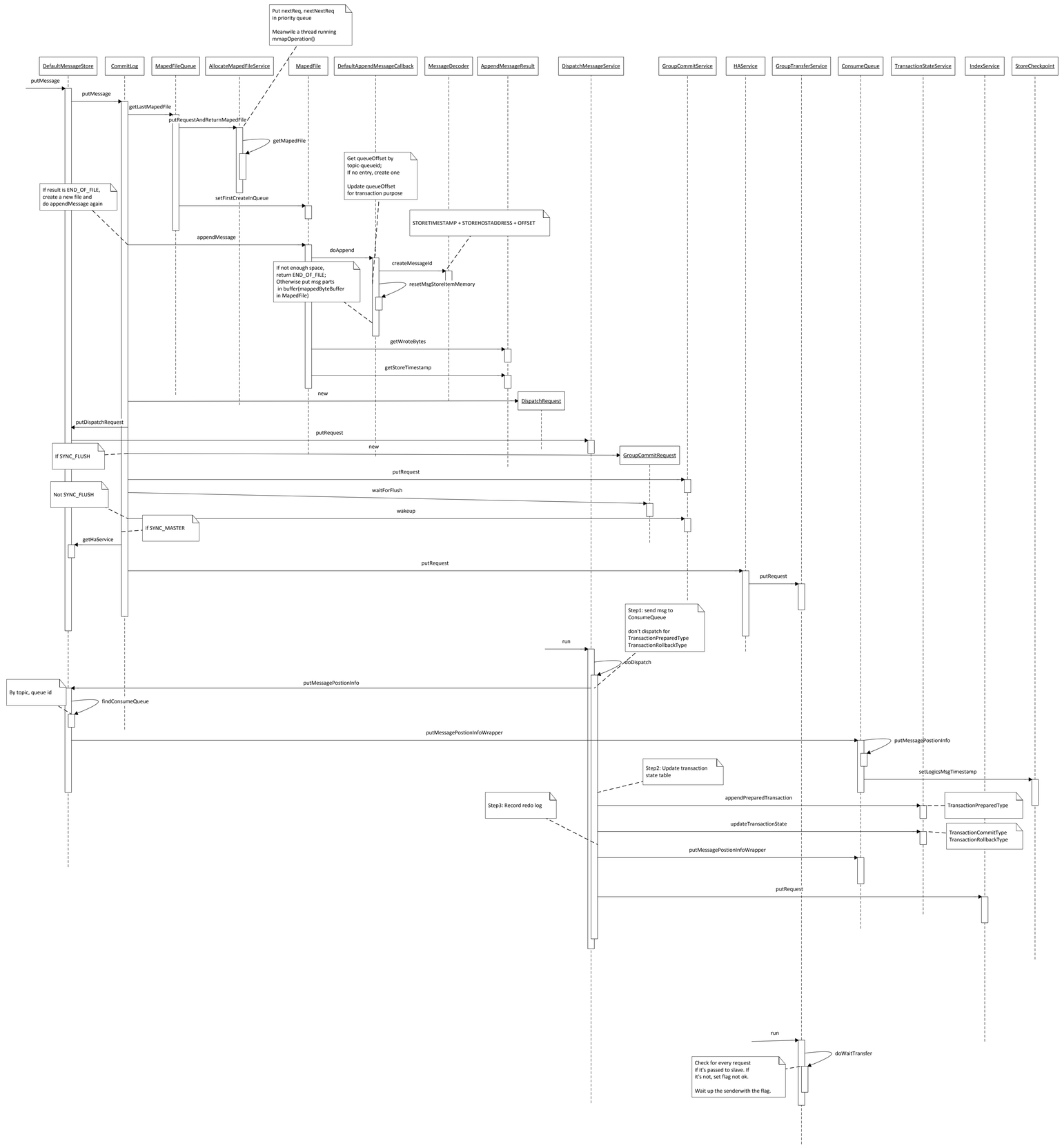

non-block and
listen to "ACCEPT"
HAService
AcceptSocketService
ServerSocketChannel
RemotingUtil
GroupTransferService
HAClient
ReadSocketService
WriteSocketService
GroupCommitRequest
DefaultMessageStore
CommitLog
MapedFileQueue
MapedFile
start
beginAccept
open
openSelector
start
start
start
run
new
HAConnection
start
start
start
run
waitForRunning
doWaitTransfer
wakeupCustomer
run
processReadEvent
Get ack offset
from slave
If nextTransferFromWhere == -1
if slaveRequestOffset == 0
nextTransferFromWhere = last file offset;
if slaveRequestOffset != 0
nextTransferFromWhere = slaveRequestOffset
run
If last time sending is done
if haven't sent heartbeat for a while
send heartbeat
else continue the sending
transferData
Construct transfer data with
configured batch size, send
getCommitLogData
getData
findMapedFileByOffset
selectMapedBuffer
transferData
Connect to master
and get max phy. offset
If it's master itself, this
function returns false,
the following work will
not be done.
run
connectMaster
reportSlaveMaxOffset
processReadEvent
dispatchReadRequest
appendToCommitLog

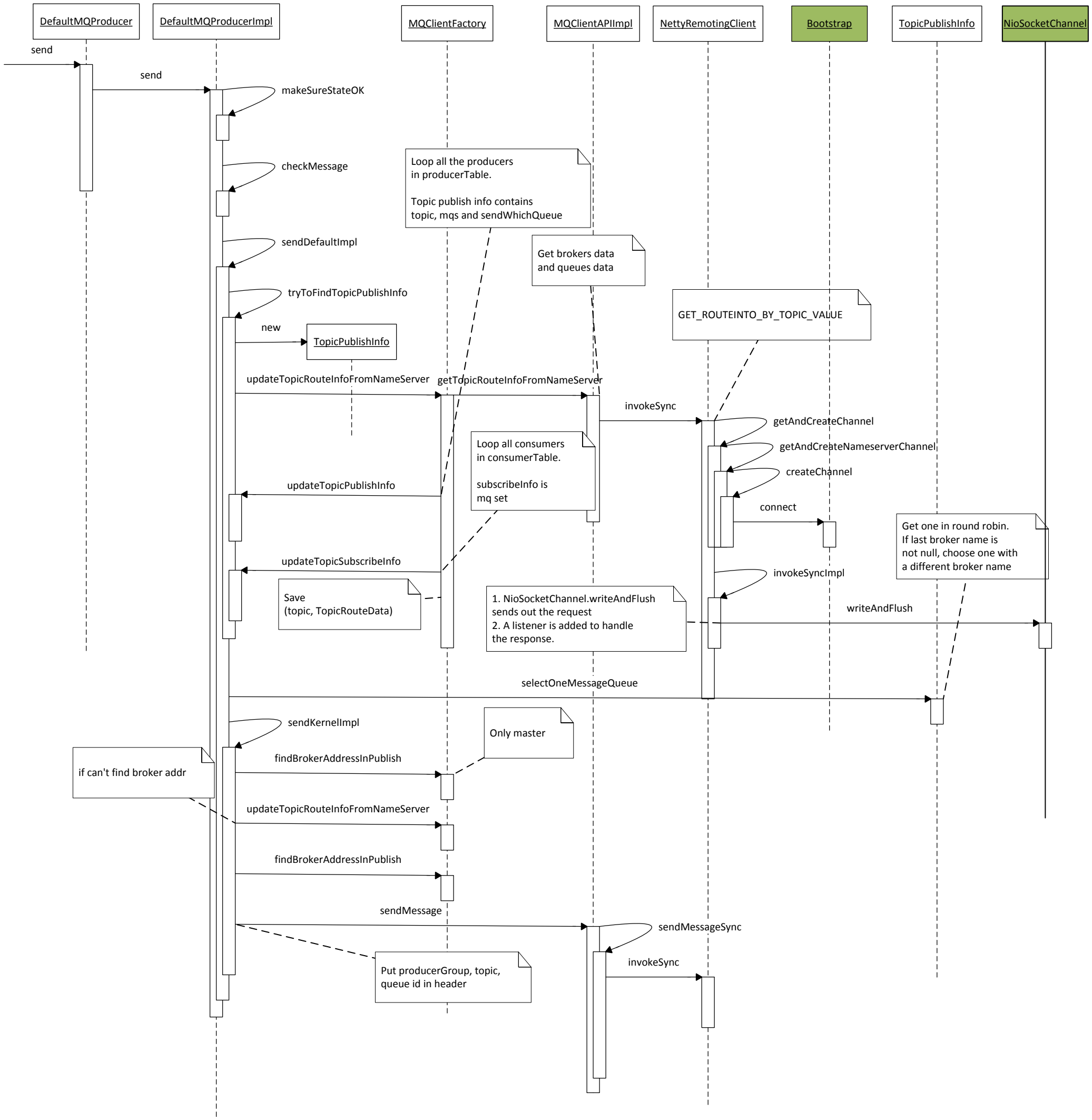

HAClient
AcceptSocketService
ReadSocketService
WriteSocketService
Connect Master
Register OP_READ
Wait for OP_READ
dispatchReadRequest
Report slave max offset
Wait for OP_ACCEPT(1s timeout)
[Not all seleted keys are accessed]
[OP_ACCEPT]
Create HAConnection and start it
Clear seleted keys
Wait for OP_READ(1s timeout)
Read slave ack offset
Update slaveAckOffset
Update slaveRequestOffset if it < 0
Wait for OP_WRITE(1s timeout)
[First time transmission]
[Slave has no data]
Send from the last phy. file
Send from the slave requested offset
[Transimission not done last time]
Continue the transimission
Send from nextTransferFromWhere with configured batch size
Update nextTransferFromWhere

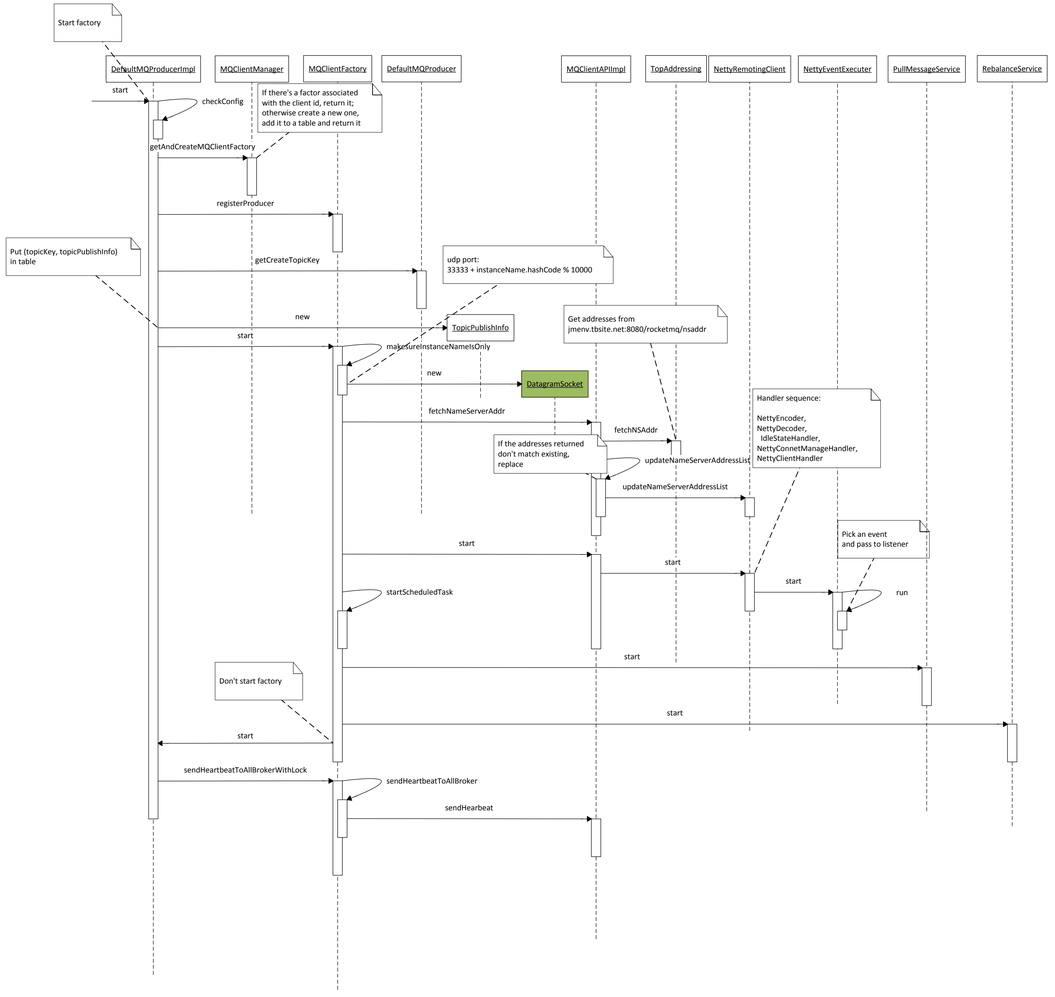

TransactionMQProducer
DefaultMQProducerImpl
TransactionExecuterImpl
MQClientAPIImpl
NettyRemotingClient
sendMessageInTransaction
sendMessageInTransaction
send
Step 1: Send "Prepared"
msg to broker by putting
(PROPERTY_TRANSACTION_PREPARED, true)
property in msg
executeLocalTransactionBranch
Step2: get local trans state
endTransaction
endTransactionOneway
invokeOneway
invokeOnewayImpl

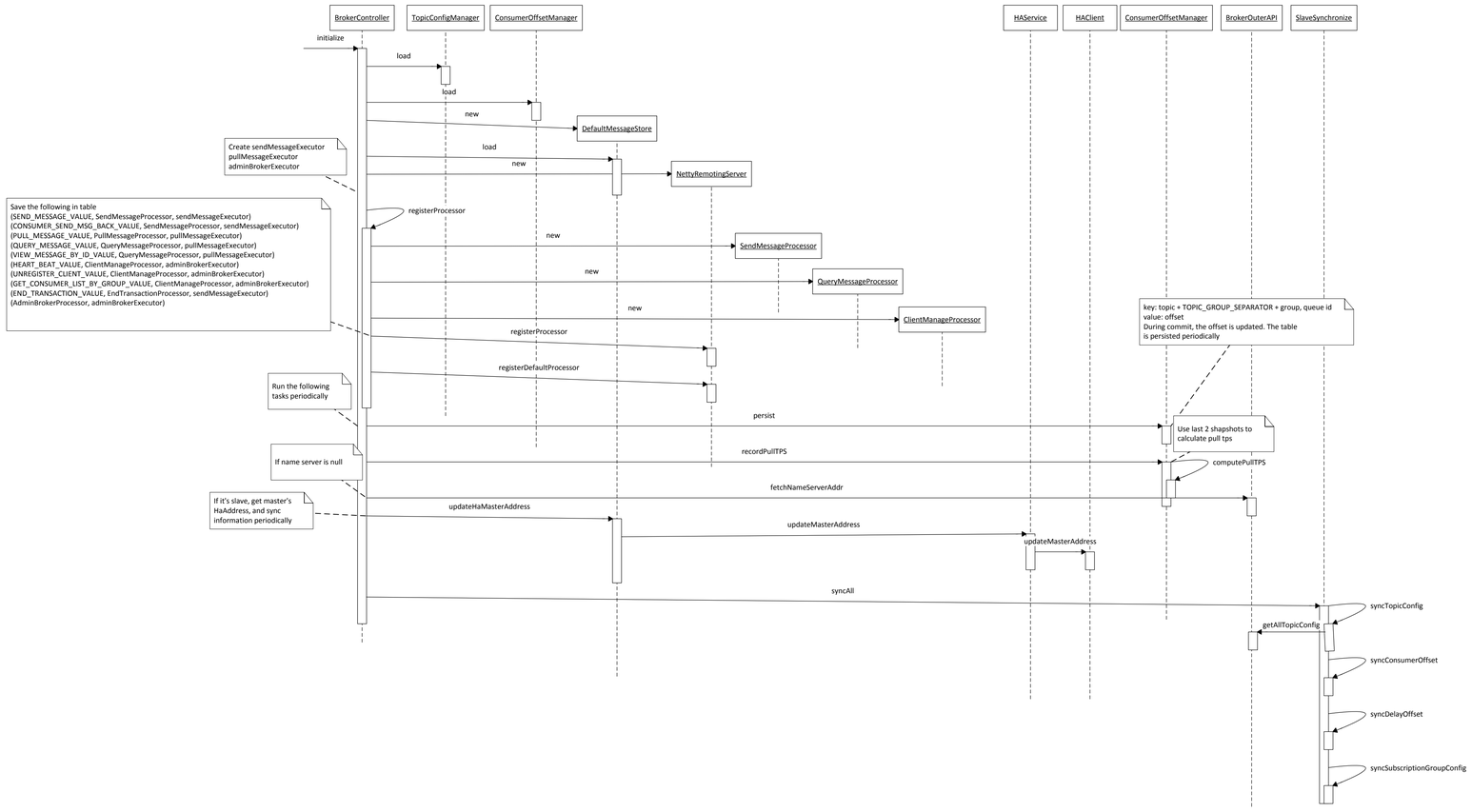

DefaultMessageStore
CommitLog
MapedFileQueue
MessageStoreConfig
TransactionStateService
ConsumeQueue
ScheduleMessageService
MessageStoreConfig
IndexService
IndexHeader
MapedFile
SelectMapedBufferResult
load
isTempFileExist
Load files under commitlog
load
load
loadConsumeQueue
ConcurrentHashMap<String/* topic */, ConcurrentHashMap<Integer/* queueId */, ConsumeQueue>>
getStorePathConsumeQueue
Iterate topic and its queues
new
ConsumeQueue
putConsumeQueue
load
load
For redo log from transaction/redolog
load
load
load
For state table from transaction\statetable
load
load
Map files under a specific directory (redo in this case) to DirectByteBuffer. Save in a map file list
parseDelayLevel
getStoreCheckpoint
new
StoreCheckpoint
Load from store\index
load
From mapped byte buffer
new
IndexFile
load
load
Check for each unit in mapfile: offset >=0 && size > 0.
Will ignore the unit doesn't hold this condition and those after it.
Set wrote & commited postion in the last file and delte the files after it.
recover
recoverConsumeQueue
Recover redo log
recover
truncateDirtyFiles
recover
Last exit ok
recoverNormally
Last exit not ok
recoverAbnormally
last exit ok
recoverStateTable
recoverStateTableNormal
topic-queueid, minPhyOffset
recoverTopicQueueTable
last exit not ok
destroy
recreateStateTable
Step 1:
Scan redo log and add "prepare" units to a set
selectMapedBuffer
release
Step 2:
Append "prepare" units to tranStateTable
appendPreparedTransaction
appendMessage

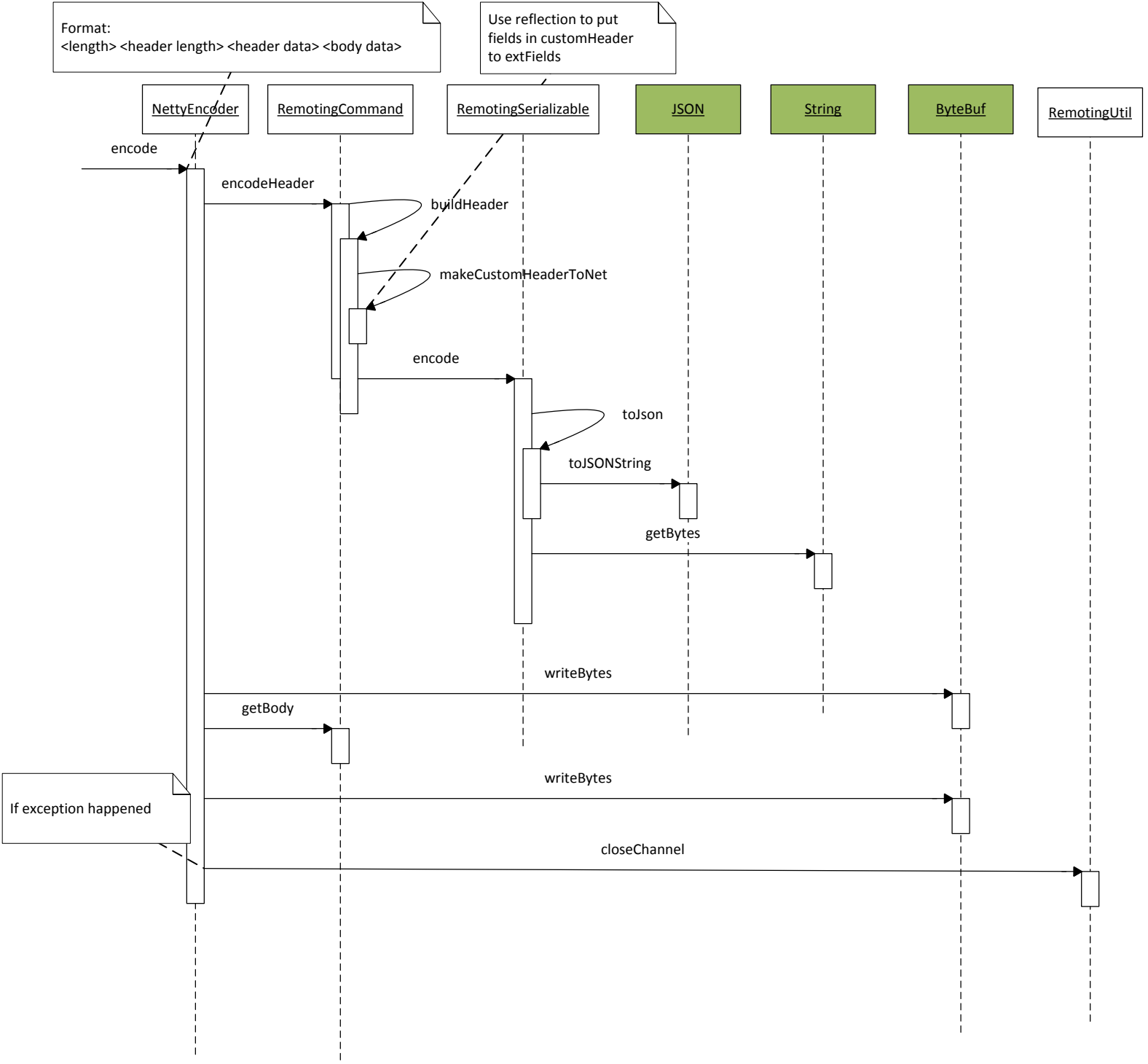

NettyServerHandler
NettyRemotingAbstract
SendMessageProcessor
DefaultMessageStore
channelRead0
processMessageReceived
processRequestCommand
Run in a thread
processRequest
sendMessage
putMessage

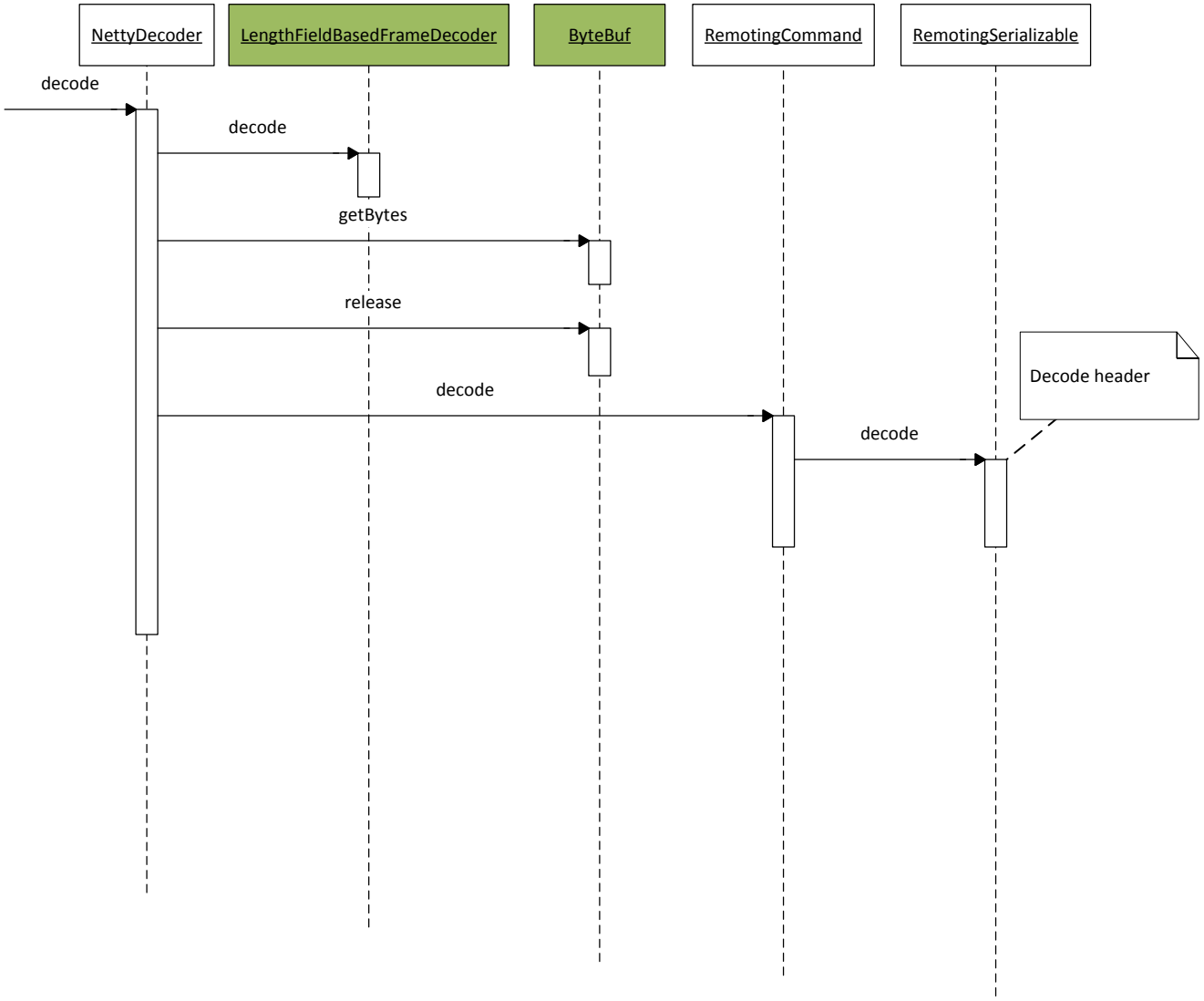

Put nextReq, nextNextReq
in priority queue
Meanwile a thread running
mmapOperation()
DefaultMessageStore
CommitLog
MapedFileQueue
AllocateMapedFileService
MapedFile
DefaultAppendMessageCallback
MessageDecoder
AppendMessageResult
DispatchMessageService
GroupCommitService
HAService
GroupTransferService
ConsumeQueue
TransactionStateService
IndexService
StoreCheckpoint
putMessage
putMessage
getLastMapedFile
putRequestAndReturnMapedFile
getMapedFile
Get queueOffset by
topic-queueid;
If no entry, create one
Update queueOffset
for transaction purpose
If result is END_OF_FILE,
create a new file and
do appendMessage again
setFirstCreateInQueue
STORETIMESTAMP + STOREHOSTADDRESS + OFFSET
appendMessage
doAppend
createMessageId
If not enough space,
return END_OF_FILE;
Otherwise put msg parts
in buffer(mappedByteBuffer
in MapedFile)
resetMsgStoreItemMemory
getWroteBytes
getStoreTimestamp
new
DispatchRequest
putDispatchRequest
putRequest
new
If SYNC_FLUSH
GroupCommitRequest
putRequest
Not SYNC_FLUSH
waitForFlush
wakeup
if SYNC_MASTER
getHaService
putRequest
putRequest
Step1: send msg to
ConsumeQueue
don't dispatch for
TransactionPreparedType
TransactionRollbackType
run
doDispatch
putMessagePostionInfo
By topic, queue id
findConsumeQueue
putMessagePostionInfoWrapper
putMessagePostionInfo
Step2: Update transaction
state table
setLogicsMsgTimestamp
Step3: Record redo log
appendPreparedTransaction
TransactionPreparedType
updateTransactionState
TransactionCommitType
TransactionRollbackType
putMessagePostionInfoWrapper
putRequest
run
doWaitTransfer
Check for every request
if it's passed to slave. If
it's not, set flag not ok.
Wait up the senderwith the flag.


DefaultMQProducer
DefaultMQProducerImpl
MQClientFactory
MQClientAPIImpl
NettyRemotingClient
Bootstrap
TopicPublishInfo
NioSocketChannel
send
send
makeSureStateOK
checkMessage
Loop all the producers
in producerTable.
Topic publish info contains
topic, mqs and sendWhichQueue
sendDefaultImpl
Get brokers data
and queues data
tryToFindTopicPublishInfo
GET_ROUTEINTO_BY_TOPIC_VALUE
new
TopicPublishInfo
updateTopicRouteInfoFromNameServer
getTopicRouteInfoFromNameServer
invokeSync
getAndCreateChannel
getAndCreateNameserverChannel
Loop all consumers
in consumerTable.
subscribeInfo is
mq set
createChannel
updateTopicPublishInfo
connect
Get one in round robin.
If last broker name is
not null, choose one with
a different broker name
updateTopicSubscribeInfo
invokeSyncImpl
Save
(topic, TopicRouteData)
1. NioSocketChannel.writeAndFlush
sends out the request
2. A listener is added to handle
the response.
writeAndFlush
selectOneMessageQueue
sendKernelImpl
Only master
if can't find broker addr
findBrokerAddressInPublish
updateTopicRouteInfoFromNameServer
findBrokerAddressInPublish
sendMessage
sendMessageSync
invokeSync
Put producerGroup, topic,
queue id in header


Start factory
DefaultMQProducerImpl
MQClientManager
MQClientFactory
DefaultMQProducer
MQClientAPIImpl
TopAddressing
NettyRemotingClient
NettyEventExecuter
PullMessageService
RebalanceService
start
checkConfig
If there's a factor associated with the client id, return it; otherwise create a new one, add it to a table and return it
getAndCreateMQClientFactory
registerProducer
Put (topicKey, topicPublishInfo) in table
getCreateTopicKey
udp port:
33333 + instanceName.hashCode % 10000
new
TopicPublishInfo
Get addresses from jmenv.tbsite.net:8080/rocketmq/nsaddr
start
makeSureInstanceNameIsOnly
new
DatagramSocket
Handler sequence:
NettyEncoder,
NettyDecoder,
IdleStateHandler,
NettyConnetManageHandler,
NettyClientHandler
fetchNameServerAddr
fetchNSAddr
If the addresses returned don't match existing, replace
updateNameServerAddressList
updateNameServerAddressList
Pick an event and pass to listener
start
start
start
run
startScheduledTask
start
Don't start factory
start
start
sendHeartbeatToAllBrokerWithLock
sendHeartbeatToAllBroker
sendHearbeat


BrokerController
TopicConfigManager
ConsumerOffsetManager
HAService
HAClient
ConsumerOffsetManager
BrokerOuterAPI
SlaveSynchronize
initialize
load
load
new
DefaultMessageStore
load
new
NettyRemotingServer
Create sendMessageExecutor
pullMessageExecutor
adminBrokerExecutor
registerProcessor
Save the following in table
(SEND_MESSAGE_VALUE, SendMessageProcessor, sendMessageExecutor)
(CONSUMER_SEND_MSG_BACK_VALUE, SendMessageProcessor, sendMessageExecutor)
(PULL_MESSAGE_VALUE, PullMessageProcessor, pullMessageExecutor)
(QUERY_MESSAGE_VALUE, QueryMessageProcessor, pullMessageExecutor)
(VIEW_MESSAGE_BY_ID_VALUE, QueryMessageProcessor, pullMessageExecutor)
(HEART_BEAT_VALUE, ClientManageProcessor, adminBrokerExecutor)
(UNREGISTER_CLIENT_VALUE, ClientManageProcessor, adminBrokerExecutor)
(GET_CONSUMER_LIST_BY_GROUP_VALUE, ClientManageProcessor, adminBrokerExecutor)
(END_TRANSACTION_VALUE, EndTransactionProcessor, sendMessageExecutor)
(AdminBrokerProcessor, adminBrokerExecutor)
new
SendMessageProcessor
new
QueryMessageProcessor
new
ClientManageProcessor
registerProcessor
registerDefaultProcessor
key: topic + TOPIC_GROUP_SEPARATOR + group, queue id
value: offset
During commit, the offset is updated. The table
is persisted periodically
Run the following
tasks periodically
persist
Use last 2 shapshots to
calculate pull tps
If name server is null
recordPullTPS
computePullTPS
fetchNameServerAddr
If it's slave, get master's
HaAddress, and sync
information periodically
updateHaMasterAddress
updateMasterAddress
updateMasterAddress
syncAll
syncTopicConfig
getAllTopicConfig
syncConsumerOffset
syncDelayOffset
syncSubscriptionGroupConfig


Format:
<length> <header length> <header data> <body data>
Use reflection to put fields in customHeader to extFields
NettyEncoder
RemotingCommand
RemotingSerializable
JSON
String
ByteBuf
RemotingUtil
encode
encodeHeader
buildHeader
makeCustomHeaderToNet
encode
toJson
toJSONString
getBytes
writeBytes
getBody
writeBytes
If exception happened
closeChannel


NettyDecoder
LengthFieldBasedFrameDecoder
ByteBuf
RemotingCommand
RemotingSerializable
decode
decode
getBytes
release
decode
decode
Decode header